MATEMATICKO-FYZIKÁLNÍ
FAKULTA
Univerzita Karlova

DIPLOMOVÁ PRÁCE

Michal Hanzeli

Anzora: Zotero klient pro Android

Katedra distribuovaných a spolehlivých systémů

Vedoucí diplomové práce: RNDr. Jan Kofroň, Ph.D.

Studijní program: Informatika

Studijní obor: Softwarové systémy

Praha 2017

Ďakujem RNDr. Janu Kofroňovi za odborné vedenie mojej práce, podnetné rady a čas, ktorý mi počas vypracovávania tejto práce venoval. Ďalej by som chcel poďakovať svojim rodičom za podporu v mojom doterajšom štúdiu.

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Univerzita Karlova má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona.

V …..... dne..... podpis

Název práce: Anzora: Zotero klient pro Android

Autor: Michal Hanzeli

Katedra / Ústav: Katedra distribuovaných a spolehlivých systémů

Vedoucí diplomové práce: RNDr. Jan Kofroň, Ph.D.

Abstrakt: Cílem práce bolo vytvořit klientskou aplikaci Anzora pro mobilní zařízení se systémem Android, která by poskytovala přístup a správu databázi zdrojů Zotero. Aplikace bude sloužit zejména jako doplněk k existující desktopové verzi a proto bude zaměřená na přístup k osobní Zotero knihovně a přílohám v online a offline módu. Poskytne také možnost úpravy existujících záznamů, přidávání nových a následnou synchronizaci s databází uloženou na serveru. Součástí práce je popis funkcí aplikace, jejího návrhu a implementace. V závěru uvádíme porovnání s existujícími aplikacemi a problémy na které jsme při jejich testováni narazili.

Klíčová slova: klient, Zotero, Android, Java

Title: Anzora: Zotero client for Android devices

Author: Michal Hanzeli

Department: Department of Distributed and Dependable Systems

Supervisor: RNDr. Jan Kofroň, Ph.D.

Abstract: The goal of this thesis was to create client application Anzora for mobile devices with Android OS that would allow access and management of the Zotero database. The application will be used mostly as an addition to an existing desktop version and therefore, it will be focused on accessing personal Zotero library and attachments in online and offline mode. It will also allow for editing existing items, adding new items, and synchronization with the server database. This thesis contains a description of functionality of the application and its design and implementation. At the end, we compare it with existing applications and present the problems we have encountered while testing them.

Keywords: client, Zotero, Android, Java

Názov práce: Anzora: Zotero klient pro Android

Autor: Michal Hanzeli

Katedra / Ústav: Katedra distribuovaných a spolehlivých systémů

Vedúcí diplomovej práce: RNDr. Jan Kofroň, Ph.D.

Abstrakt: Cieľom práce bolo vytvoriť klientsku aplikáciu Anzora pre mobilné zariadenia so systémom Android, ktorá by poskytovala prístup a správu databáze zdrojov Zotero. Aplikácia bude slúžiť hlavne ako doplnok k existujúcej desktopovej verzii a preto bude zameraná na prístup k osobnej Zotero knižnice a prílohám v online a offline móde. Poskytne tiež možnosť úpravy existujúcich záznamov, pridávania nových a následnú synchronizáciu s databázou uloženou na serveri. Súčasťou práce je popis funkcií aplikácie a jej návrhu a implementácie. V závere uvádzame porovnanie s existujúcimi aplikáciami a problémy, na ktoré sme pri ich testovaní narazili.

Kľúčové slová: klient, Zotero, Android, Java

Obsah

1. Úvod ... 3

2. Zotero ... 4
   2.1. Užívateľské prostredie Zotero pre webový prehliadač ... 4
   2.2. Prístup do osobnej knižnice ... 5
   2.3. Zotero Web Api ... 5

3. Analýza ... 7
   3.1. Špecifikácia funkčných požiadaviek ... 7
   3.2. Autentifikácia užívateľa ... 7
   3.3. Komunikácia so serverom ... 8
   3.4. Lokálna databáza ... 9
   3.5. Synchronizácia ... 10
   3.6. Design užívateľského prostredia ... 11
   3.7. Návrh aplikácie ... 12
   3.8. Výber cieľovej verzie systému Android ... 13

4. Popis aplikácie a jej funkcionality ... 14
   4.1. Prístup do osobnej knižnice ... 14
   4.2. Prehliadanie a práca s knižnicou ... 14
   4.3. Vytvorenie webovej položky ... 15
   4.4. Synchronizácia ... 16

5. Popis implementácie ... 17
   5.1. Použité technológie ... 17
   5.2. Základ aplikácie ... 17
   5.3. AnzoraApplication ... 18
   5.4. MainActivity ... 19
   5.5. LibraryActivity ... 20
   5.6. ItemInfoActivity ... 22
   5.7. Library ... 24
   5.8. Synchronization ... 27

6. Porovnanie existujúcich aplikácií ... 30
   6.1. Zojo ... 30
   6.2. Zotable ... 31
   6.3. Zed Lite ... 32

7. Záver ... 33

Zoznam použitej literatúry ... 34

8. Prílohy ... 35
   8.1. Užívateľská dokumentácia ... 35
      8.1.1. Prihlasovacia obrazovka ... 35
      8.1.2. Obrazovka s obsahom knižnice ... 36

8.1.3. Obrazovka s informáciami o položke 38
8.1.4. Zdieľanie webovej stránky 39
8.2. Inštalácia aplikácie 40
8.3. Obsah priloženého prenosového média 40

1. Úvod

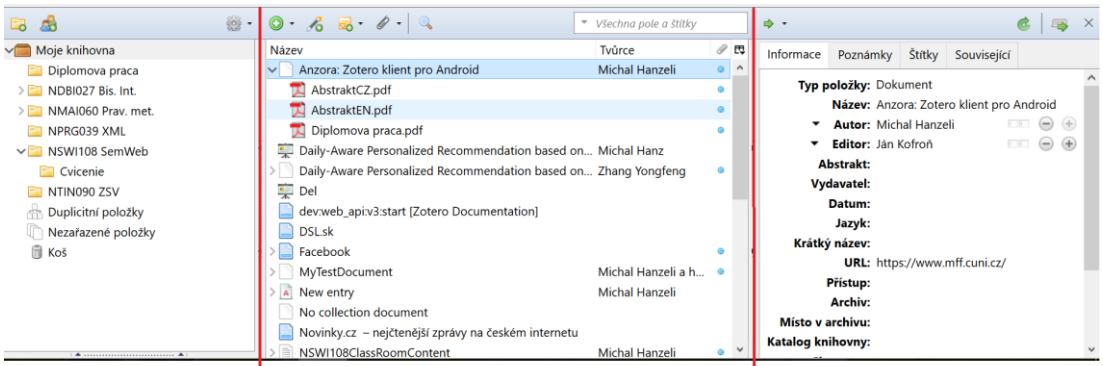
Vzdelávanie a iné odborné činnosti závisia na práci s odbornými dokumentmi a vedeckými materiálmi (ako sú PDF súbory). To zo sebou prináša potrebu spravovať tieto materiály a vytvárať si akúsi osobnú knižnicu. Na tento účel bolo vytvorených mnoho programov, ako sú napríklad BibDesk [1], ReadCube [2], Papers [3] a iné. Jedným z nich je aj open-source manažér Zotero [4], ktorého hlavným zameraním je ukladať a spravovať odborné dokumenty získané z webových stránok a iných zdrojov. Zároveň umožňuje synchronizáciu uložených dát s ich zálohou na Zotero serveri a prostredníctvom Web API [5] k nim pristupovať z rôznych zariadení.
Cieľom práce bolo navrhnúť a implementovať aplikáciu s názvom Anzora pre systém Android, ktorá by umožňovala prístup a správu osobnej Zotero knižnice. Slúžiť bude hlavne ako doplnok k existujúcej desktopovej verzii Zotero Standalone a preto bude skôr zameraná na prehliadanie obsahu tejto databázy a uložených príloh (PDF súborov). Užívateľ bude pomocou tejto aplikácie schopný prehliadať obsah knižnice, zobraziť požadovanú položku a otvoriť jej prílohu. Zároveň poskytne aj možnosť jednoducho upravovať existujúce položky a pridávať nové položky. Zotero je dostupné aj ako rozšírenie pre prehliadač Mozilla Firefox, kde slúži pre pridávanie odkazov webových stránok do databázy. Chceli sme preto toto chovanie v určitej miere preniesť aj do našej mobilnej verzie, keďže v súčasnosti je bežné prehliadanie webových stránok na mobilnom zariadení. Po vykonaní zmien v lokálnej databáze bude mať užívateľ možnosť nahrať tieto zmeny na server a udržiavať tak svoju knižnicu synchronizovanú naprieč zariadeniami. Vďaka vytváraniu lokálnej databázy bude aplikácia schopná pracovať v „online“ aj „offline“ móde. Anzora bude využívať užívateľské prostredie podobné desktopovej verzii a uľahčí sa tým jej použitie. Zároveň však ponúkne responzívny design, ktorý podporuje rôzne veľkosti displeja a reaguje aj na zmenu jeho orientácie (portrait/landscape mode).

Prvá kapitola sa bude zaoberať stručným predstavením manažéra Zotero a jeho hlavných funkcií. V druhej kapitole uvedieme analýzu požiadaviek na našu aplikáciu a navrhneme spôsob riešenia. V ďalšej, tretej kapitole, špecifikujeme našu aplikáciu a popíšeme jej funkcie. Štvrtá kapitola bude obsahovať popis implementácie aplikácie a v poslednej, piatej kapitole, porovnáme existujúcich Zotero klientov pre Android. V závere zhodnotíme našu prácu a jej ciele.

2. Zotero

Zotero je open-source citačný manažér, ktorý bol vyvinutý na univerzite George Mason Univerzity, Center for History and New Media v aktuálnej verzii 4.0.29 [6]. Umožňuje ukladať a spravovať články a dokumenty (ako sú PDF súbory) hlavne z oblasti vedeckých publikácií. Hlavnými funkciami sú integrácia do webových prehliadačov (Firefox, Chrome a Safari) a textových procesorov (Microsoft Word, LibreOffice, OpenOffice a NeoOffice), online synchronizácia a generovanie citácií, poznámok pod čiarou a bibliografií. Existuje aj ako samostatný program Zotero Standalone pre systém Windows. Jeho užívateľské prostredie je zhodné s prostredím doplnku pre webový prehliadač a preto si stručne popíšeme prostredie pre prehliadač.

2.1. Užívateľské prostredie Zotero pre webový prehliadač



Obrázok 1: Prostredie doplnku Zotero

Prostredie doplnku Zotero (Obrázok 1) pre webový prehliadač je vertikálne rozdelené na tri časti. Ľavý stĺpec obsahuje užívateľovu kompletnú knižnicu („Moje knihovna“) a jednotlivé kolekcie, ktoré sú jej podmnožinou. Knižnica obsahuje všetky súbory a prílohy, ktoré boli do nej uložené, stiahnuté alebo vpísané. Stredný stĺpec ukazuje položky v kolekcii, ktorá je práve zvolená v pravom stĺpci. Pravý stĺpec ukazuje informácie o položke, ktorá je vybraná v strednom stĺpci a zároveň umožňuje ich editáciu (sú to napríklad jej názov, typ, tvorcovia, dátum modifikácie, prílohy a iné). Pomocou „drag-and-drop“ gesta je možné presúvať vybranú položku medzi kolekciami alebo klávesou „Delete“ z knižnice zmazať. Zároveň je možné pridávať poznámky pre jednotlivé položky v pravom informačnom stĺpci.

2.2. Prístup do osobnej knižnice

Užívateľ, ktorý si prostredníctvom webového formuláru vytvorí Zotero účet, získa možnosť vytvárania osobnej online knižnice, ktorú potom spravuje pomocou nástrojov uvedených v predošlej kapitole. Zotero využíva pre prístup ku knižniciam protokol Oauth (v súčasnosti verziu 1.0a), ktorý poskytuje bezpečnú autorizáciu. Ten funguje na nasledovnom princípe. Každá klientska aplikácia, ktorá chce komunikovať s online databázou, musí byť zaregistrovaná pomocou webového formulára dostupného na stránkach zotero.org. Po registrácii je aplikácii pridelený unikátny kľúč (*clientKey*) a heslo (*clientSecret*), ktoré sú potom využité pri prihlasovaní užívateľa do aplikácie. Tá prebieha v troch krokoch:

1. Odoslanie *clientKey* a *clientSecret* na URL, kde sa overia prihlasovacie údaje aplikácie a získa sa dočasný komunikačný kľúč.
2. Užívateľ je prostredníctvom webového prehliadača presmerovaný na URL s odkazom na dočasný komunikačný kľúč, kde ho overí svojimi prihlasovacími údajmi.
3. Po úspešnom overení užívateľa je vrátený unikátny kľúč užívateľa (*userID*) a permanentný komunikačný kľúč (*userSecret*) . Tieto údaje sú potom používané pri všetkých budúcich požiadavkách o dáta pre tohto užívateľa.

2.3. Zotero Web Api

Zotero poskytuje prístup k online databáze pomocou jednoduchého webového rozhrania využívajúceho HTTPS žiadosti. Základom každej žiadosti je adresa:

```
https://api.zotero.org
```

Žiadosti o dáta z osobnej knižnice daného užívateľa začínajú s prefixom:

```
/users/<userID>
```

kde <userID> je unikátny kľúč daného užívateľa získaný po jeho úspešnej autentifikácii popísanej v predošlej kapitole. Každá žiadosť taktiež musí obsahovať permanentný komunikačný kľúč (*userSecret*). Príklady žiadostí o konkrétne dáta:

Žiadosť o všetky položky v knižnici:
https://api.zotero.org/users/2807516/items?key=cyT8c3nkzYY4JzWwvELX4NPJ

Žiadosť o všetky kolekcie v knižnici:
https://api.zotero.org/users/2807516/collections?key=cyT8c3nkzYY4JzWwvELX4NPJ

Žiadosť o položky v špecifickej kolekcii v knižnici:
https://api.zotero.org/users/2807516/collections/WCVP737K/items?key=cyT8c3nkzYY4JzWwvELX4NPJ

3. Analýza

Vytváranie aplikácie znamenalo v prvom rade nadefinovanie hlavných funkčných požiadaviek, od ktorých by sa odvíjal celý jej vývoj a použité technológie. Zároveň sme sa snažili použiť také doplnkové balíčky, ktoré by rozumne pokrývali naše potreby a nezaťažovali by zbytočne svojou veľkosťou distribučný balíček . Pri vývoji sme vychádzali aj z predpokladu, že užívateľ má skúsenosti s používaním klasickej verzie Zotera, keďže táto aplikácia bude využívaná ako jej doplnok. Riešenie sme si rozdelili na niekoľko bodov, ktoré si v tejto kapitole uvedieme spolu s použitými postupmi.

3.1. Špecifikácia funkčných požiadaviek

Aplikácia z funkčného hľadiska musí spĺňať nasledovné požiadavky:

1. Užívateľ sa do osobnej knižnice prihlasuje len pri prvom použití aplikácie a po manuálnom odhlásení
2. Užívateľ bude mať po prihlásení a stiahnutí knižnice prístup k položkám z databázy
3. Aplikácia musí fungovať v „online“ aj „offline“ móde – to znamená, že bez prístupu k internetovému pripojeniu musí mať užívateľ prístup k položkám knižnice a k už stiahnutým prílohám.
4. Užívateľ bude schopný v obmedzenej miere editovať existujúce položky v knižnici.
5. Užívateľ bude schopný manuálne pridávať nové položky do knižnice.
6. Užívateľ bude schopný pridávať položky typu webová stránka prostredníctvom zdieľania webovej stránky z aplikácie webového prehliadača.
7. Aplikácia bude poskytovať možnosť synchronizácie lokálnych a vzdialených zmien.

3.2. Autentifikácia užívateľa

Aplikácia bude určená pre prístup do osobnej knižnice užívateľa a preto bolo nutné, ako prvé, navrhnúť autentifikáciu užívateľa. V predošlej kapitole sme si popísali, akým spôsobom je možné postupovať pri získavaní prístupu do online knižnice. Vy-

žadovali sme, aby bolo potrebné vykonať prihlásenie len jedenkrát pri prvom použití aplikácie. Tento postup je bežným pri mobilných aplikáciách, ktoré pristupujú k osobným online databázam, keďže je mobilné zariadenie zväčša využívané vlastníkom tohto zariadenia a už pre samotný vstup do zariadenia je nutné ho najskôr odblokovať.

Štandardné prihlasovanie pomocou OAuth protokolu nám teda vyhovovalo. Aplikácia pri prvom prihlásení užívateľa získa jeho údaje (užívateľova identifikácia a permanentný komunikačný kľúč), ktoré si uloží a následne využije pri každej komunikácii so serverom. Týmto spôsobom sme pokryli prvý bod z funkčných požiadaviek.

3.3. Komunikácia so serverom

Ďalšou z požiadaviek bol prístup k položkám z užívateľovej knižnice. V predošlej kapitole sme si uviedli, že Zotero umožňuje pristupovať k online databáze pomocou HTTPS žiadostí. Využili sme tento princíp, keďže Java poskytuje triedy pre podporu HTTP komunikácie v rámci balíčka java.net.

Zotero podporuje tri formáty pre dáta, ktoré sú obsiahnuté v odpovedi na žiadosť o čítanie. Prvým je bibliografický formát slúžiaci na výpis informácií o položkách v knižnici pomocou určitého citačného štýlu (štýl je možné nastaviť pomocou parametrov HTTPS žiadosti). Tento formát nám nevyhovoval, pretože je spracovanie takýchto dát náročné a zároveň je možné posielať len HTTPS žiadosti o položky a nie kolekcie. Druhým je formát Atom [7], ktorý je určený pre zdieľanie aktualizačných správ prostredníctvom webových informačných kanálov. Tento formát nám tiež nevyhovoval, pretože prenášané dáta obsahujú formátovanie pomocou html značiek vhodných pre zobrazenie vo webovom prehliadači. V našej aplikácii by sme tieto značky zahadzovali, pretože štýl zobrazenia dát bude daný samotnou aplikáciou. Prenášali by sme tak zbytočne veľa dát, ktoré nevyužijeme. Posledným je formát JSON, ktorý nám už vyhovoval, pretože je jednoduchý na spracovanie a prenášané dáta obsahujú len informácie potrebné pre vytvorenie lokálnej knižnice. Na to, aby sme mohli JSON formát spracovávať, sme potrebovali vhodný „parser“. V rámci našej aplikácie nám postačovalo jednoduché čítanie a zapisovanie do JSON objektov. Zvolili sme preto balíček org.json, ktorý je priamo súčasťou Android SDK.

Dôležitým aspektom komunikácie bolo potrebné oddeliť tieto operácie do pozadia aplikácie. Hlavným dôvodom bolo, že užívateľské prostredie beží v jednom vlákne. Ak by sme v rámci neho vykonávali aj operácie spojené s komunikáciou, spôsobovalo by to nestabilitu aplikácie. Vhodným preto bolo použitie *AsyncTask* triedy, ktorá poskytuje vykonávanie operácií na pozadí a návrat výsledkov do UI vlákna bez nutnosti zložitej manipulácie s vláknami.

3.4. Lokálna databáza

Aplikácia musí fungovať aj bez pripojenia k internetu a to znamenalo vytvorenie lokálnej databázy pre knižnicu. Tá by obsahovala všetky položky, ktoré sú uložené na Zotero serveri. Pre udržanie perzistentných dát ponúka platforma Android niekoľko možností.

Prvou je použitie triedy *SharedPreferences*, ktorá ponúka jednoduchý koncept ukladania a získavania dvojíc „kľúč-hodnota“. Umožňuje uloženie primitívnych dátových typov boolean, int, long, float a string. Táto metóda by nám pre vytvorenie celej lokálnej databázy nevyhovovala práve kvôli ukladaniu dát do dvojíc. Pre každú položku v knižnici by sme museli vytvárať unikátne kľúče pre uloženie jednotlivých informácií. To by bolo veľmi neefektívne. Triedu *SharedPreferences* by sme však v našej aplikácii využili na uloženie jednoduchých informácií, ako sú napríklad prihlasovacie údaje užívateľa.

Ďalšou možnosťou bolo ukladanie dát do súborov. Pre vytvorenie knižnice by bola táto metóda nevýhodná, pretože budeme vykonávať časté operácie čítania a zapisovania relačných dát v rámci behu aplikácie. Na to by sme museli vytvoriť špeciálny systém, ktorý by takéto operácie umožňoval. Túto metódu ale použijeme na ukladanie stiahnutých príloh.

Najvhodnejšou možnosťou bude tretia metóda, a to relačná databáza SQLite. Táto databáza bude v aplikácii jednoducho prístupná a pomocou SQL výrazov nad ňou budeme vykonávať jednotlivé operácie. Výhodou zároveň bude, že táto databáza nebude prístupná iným aplikáciám a bude tak bezpečným spôsobom vytvorenia lokálnej knižnice.

Cieľom práce nebolo navrhnúť plnohodnotného klienta, preto sme sa rozhodli, že aplikácia bude o každej položke v knižnici zobrazovať len určitú podmnožinu

informácií v porovnaní s „desktopovou“ verziou. Bude zahŕňať názov položky, jej typ, zoznam tvorcov, url adresu, dátum a čas modifikácie a zoznam príloh. Informácie o položke budú editovateľné a zároveň bude možné vytvoriť položku, alebo ju zmazať.

Prílohy, ktoré budú naimportované do online databázy bude možné stiahnuť do zariadenia. Stiahnutie vybranej prílohy sa vykoná len raz pri prvom pokuse o jej zobrazenie a to do konkrétneho adresára vyhradeného pre našu aplikáciu. Každé ďalšie zobrazenie už nebude vyžadovať opätovné stiahnutie prílohy, ale len jej otvorenie z lokálneho adresára. Takto bude užívateľ môcť pracovať s aplikáciou aj bez pripojenia na internet.

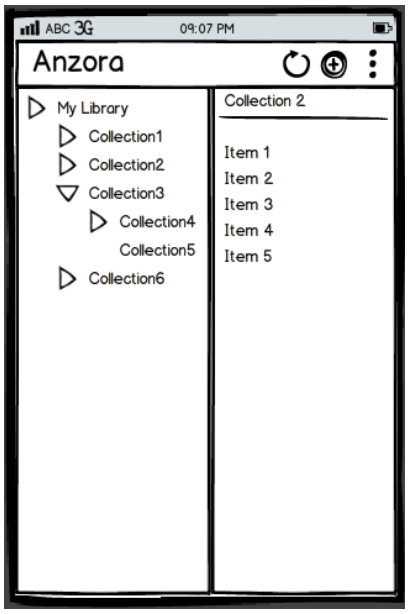
3.5. Synchronizácia

Úpravy na položkách, ktoré užívateľ vykoná v našej aplikácii a na inom zariadení, bude nutné synchronizovať. To bude znamenať stiahnutie a aplikovanie vzdialených zmien a nahratie lokálnych zmien. Keďže sa naša aplikácia bude zameriavať na osobnú databázu, budeme predpokladať, že k danej knižnici bude pristupovať len daný užívateľ a bude mať tak prehľad o vykonaných úpravách. Rozhodli sme sa preto o agresívny prístup k synchronizácii. To znamená, že ak dôjde ku konfliktnej situácii, kde na serveri bude nahraná aktualizovaná položka a tá istá položka bude upravená aj v lokálnej knižnici, na lokálnu položku sa aplikujú zmeny stiahnuté zo serveru. Užívateľ, vždy po dokončení synchronizácie, dostane stručný výpis zmien: zoznam nových a (konfliktných a bezkonfliktných) updatovaných položiek.

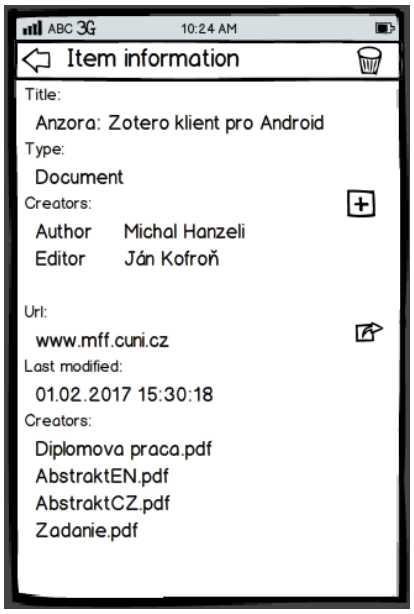
Zotero databáza funguje na princípe číselného verzovania knižnice a jej objektov (kolekcií, položiek, príloh,...). Toto verzovanie funguje tak, že vždy, ak dôjde k nahratiu upravených objektov, je inkrementovaná verzia knižnice a táto nová verzia je daným objektov priradená. Zotero web api poskytuje možnosť vyžiadania vzdialených zmien na základe čísla verzie databázy. Klient tak jednoducho zistí, či má aktuálne dáta. My túto vlastnosť využijeme a budeme si po každej synchronizácii túto verziu pamätať. Pri každej ďalšej synchronizácii pošleme žiadosť len o nové dáta (objekty s verziou vyššou ako je nami zapamätaná), ktoré následne spracujeme. Získame tak efektívnu synchronizáciu hlavne z hľadiska množstva prenášaných dát.

3.6. Design užívateľského prostredia

Pri návrhu užívateľského prostredia sme sa snažili priblížiť „desktopovej“ verzii, keďže jej prostredie je prehľadné a užívatelia programu Zotero sú na neho zvyknutý. Zároveň by sme využili moderné prvky využívané v mobilných aplikáciách. Použitie známeho prostredia a ovládacích prvkov by tak užívateľovi zjednodušilo prechod medzi „desktopovou“ a našou mobilnou verziou. Tento fakt by tak prispel k celkovému užívateľskému komfortu pri používaní aplikácie. Vytvorili sme nasledujúci návrh prostredia.



Obrázok 2: Obrazovka s knižnicou



Obrázok 3: Obrazovka s položkou

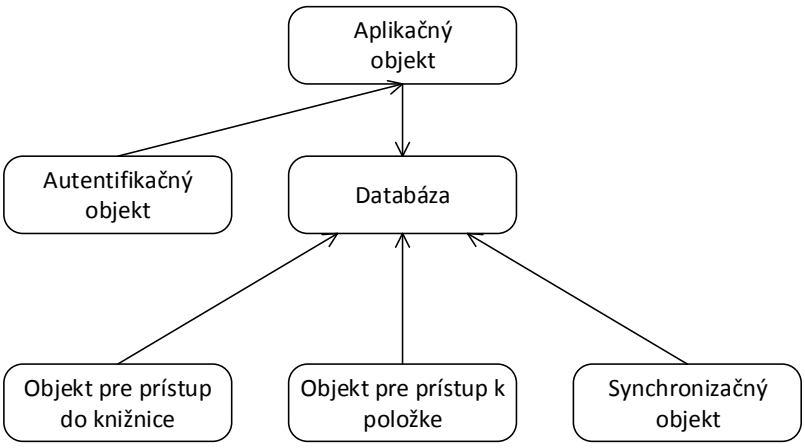
Obrazovka s knižnicou (Obrázok 2) by bola rozdelená na dve časti. V ľavej časti by sa nachádzal rozbaľovací strom s kolekciami. Pravá časť by obsahovala zoznam položiek patriacich do zvolenej kolekcie. Týmto by sme dosiahli podobnosť s ľavým a stredným stĺpcom v desktopovej verzii popísanej v predošlej kapitole. Po kliknutí na položku by sa zobrazila nová obrazovka s informáciami o danej položke (Obrázok 3). Bude podobná pravému stĺpcu z desktopovej verzie a užívateľ tu bude schopný informácie editovať(zmeniť názov položky, jej tvorcov a adresu URL). Zároveň sa tu bude nachádzať zoznam s dostupnými prílohami danej položky.

Vytváranie novej položky typu webová stránka by fungovalo prostredníctvom zdieľania z aplikácie Chrome. Po kliknutí na položku „Zdieľať“ v menu prehliadača a zvolení aplikácie Anzora sa otvorí priamo obrazovka s informáciami o vytváranej

položke. Tu sa automaticky vyplní jej názov, typ a URL podľa dát zo zdieľanej stránky.

Užívateľské prostredie bude navrhnuté tak, aby využívalo orientácie zariadenia. To bude znamenať, že aplikácia prispôsobí rozloženie jednotlivých grafických komponent tak, aby vyhovovali danej orientácii. Zároveň však musíme počítať aj s rôznymi rozmermi obrazoviek a prispôsobiť veľkosti komponent relatívne k veľkosti konkrétnej obrazovky.

3.7. Návrh aplikácie



Obrázok 4: Kostra aplikácie

Dôležitou časťou návrhu aplikácie bolo vytvorenie vhodnej kostry (Obrázok 4) a celkového rozdelenia na jednotlivé funkčné celky. Prvým celkom bola autentifikácia užívateľa do online databázy. Vytvorili sme autentifikačný objekt, ktorý bude spravovať overovanie a zároveň bude tvoriť vstupný prvok do aplikácie. Pri každom spustení aplikácie sa tento objekt načíta ako prvý a zistí stav prihlásenia. Ďalej bolo potrebné vytvoriť databázový objekt, ktorý by sa postaral o uloženie lokálnej knižnice. Zároveň by bolo jeho úlohou poskytovať rozhranie pre prístup do databázy. Toto rozhranie by pokrylo všetky operácie využívané aplikáciou, hlavne získavanie záznamov a ich úpravu. Na túto databázu by boli naviazané dva objekty, ktoré by získané záznamy poskytovali prezentačnej vrstve a vykonávali by logiku zobrazovania. Prvý objekt (Objekt pre prístup do knižnice) by sa staral o získavanie zoznamu kolekcií a k ním príslušných položiek. Úlohou druhého objektu (Objekt pre prístup k položke) by boli operácie spojené so zobrazovaním a úpravou informácií

o konkrétnej položke z knižnice. Pre zaistenie procesu synchronizácie sme vytvorili ďalší samostatný synchronizačný objekt. Ten bude taktiež priamo komunikovať s databázou a vykonávať na pozadí aplikácie všetky operácie spojené aplikovaním lokálnych a vzdialených zmien. Poslednou časťou bolo vytvorenia aplikačného objektu, ktorý by zastrešoval celú aplikáciu. Jeho hlavnou úlohou bude udržiavať stav aplikácie pri konfiguračných zmenách (ako je napríklad zmena orientácie zariadenia). Týmto návrhom sme dostali kostru celej aplikácie, na ktorú sa naviaže grafické rozhranie a iné pomocné objekty.

3.8. Výber cieľovej verzie systému Android

Pri výbere podporovaných verzií systému Android sme sa orientovali podľa dostupného testovacieho hardwaru. Na základe toho bolo možné explicitne špecifikovať cieľovú verziu systému Android, ktorú budeme podporovať. Zvolená bola teda verzia 6.0. Táto verzia je zároveň podľa súčasného stavu nainštalovaná na najväčšom počte zariadení. Podľa dostupných zdrojov [8] je relatívny počet zariadení s touto verziou 31%. Spolu so špecifikovaním minimálnej verzie na 5.0 získame kompatibilitu na veľkom počte zariadení a zároveň tak zabezpečíme, aby nedošlo k porušeniu kompatibility na zariadeniach so staršou verziou.

4. Popis aplikácie a jej funkcionality

Táto kapitola bude obsahovať detailnejší popis našej aplikácie a jej funkcionality. Detailný popis užívateľského prostredia a manuál aplikácie je uvedený v prílohe v časti užívateľská dokumentácia.

4.1. Prístup do osobnej knižnice

Aplikácia podporuje prístup do osobnej Zotero knižnice. Tento prístup je autentifikovaný, čo znamená, že po spustení aplikácie je užívateľ presmerovaný na webovú stránku Zotera. Tu zadá svoje prihlasovacie meno a heslo a povolí tak prístup Anzore do svojej online knižnice. Tento prístup bude platný pri každom ďalšom spustení aplikácie až do chvíle, pokiaľ sa užívateľ nerozhodne manuálne odhlásiť.

4.2. Prehliadanie a práca s knižnicou

Hlavnou funkciou aplikácie Anzora je umožniť užívateľovi prehliadanie jeho knižnice. Po prihlásení a iniciálnej synchronizácii sa zo serveru stiahnu všetky kolekcie a položky a uložia sa do lokálnej databázy. Následne sa tieto dáta zobrazia v samostatnej obrazovke zobrazujúcej obsah knižnice.

Kolekcie majú charakter stromovej štruktúry, kde každá kolekcia má ľubovoľný počet podkolekcií, ktoré majú zase svoje podkolekcie atď. V aplikácii je tento princíp použitý tak, že z množiny všetkých kolekcií v knižnici sa pri ich zobrazení vytvorí rozbaľovací strom. Koreň je nazvaný „Moja knižnica“ a po rozkliknutí sa pod ním zobrazí počiatočná vrstva kolekcií položiek. Po rozkliknutí niektorej z kolekcie sa zobrazia jej podkolekcie.

Položky sú zobrazované na základe ich príslušných kolekcií, do ktorých patria. To znamená, že po kliknutí na niektorú z kolekcií, je v samostatnej časti obrazovky zobrazený zoznam s položkami do nej patriacimi. Ak je vybraný koreň „Moja knižnica“, zobrazí sa zoznam so všetkými položkami v knižnici. Položky, ktoré nemajú priradenú kolekciu, sú zobrazené v rámci kolekcie nazvanej „Nezaradené“.

Po vybratí niektorej zo zobrazených položiek sa otvorí nová obrazovka, ktorá obsahuje informácie o vybranej položke: názov, typ, zoznam tvorcov, časová značka poslednej modifikácie, URL a zoznam príloh. Na tejto obrazovke následne môže

užívateľ editovať názov, tvorcov a URL alebo celú položku zmazať. Po kliknutí na tvorcu sa zobrazí dialógové okno, kde je možné upraviť tvorcu – jeho typ, meno a priezvisko. Úprava mena a priezviska je možná dvoma spôsobmi: meno a priezvisko je spojené do jedného záznamu, alebo rozdelené zvlášť do dvoch záznamov pre meno a priezvisko. Typ tvorcu sa vyberá z dostupných variant podľa typu položky. Ako príklad uvedieme: položka typu „dokument“ obsahuje tvorcov typu: „autor“, „prispievateľ“, „editor“, „autor revízie“ a „prekladateľ“. Zoznam je možné ďalej upraviť pridávaním alebo mazaním jednotlivých tvorcov. Záznam s url adresou sa po kliknutí na príslušné tlačidlo otvorí vo webovom prehliadači. Pri každej úprave niektorého zo záznamov sa automaticky upravuje časová značka na lokálny čas vo formáte „dd.mm.yyyy mm:hh:ss“. Posledným záznamom je zoznam príloh patriaci danej položke. Každá príloha má priradený typ podľa toho, akým spôsobom bola do knižnice pridaná.

- Importovaný súbor a importovaná adresa URL – v tomto prípade sa jedná o kópie príloh uložených na serveri. Prílohy týchto typov sú pri prvom pokuse o otvorenie stihnuté do aplikácie a následne je možné ich otvoriť. Ak sa jedná o importovaný súbor, je na základe jeho MIME typu poskytnutý zoznam aplikácii nainštalovaných v zariadení, ktoré daný typ dokážu otvoriť. Ak sa jedná o url adresu súbor s html stránkou sa otvorí vo webovom prehliadači.
- Nalinkovaná url adresa - táto adresa sa otvorí vo webovom prehliadači.
- Nalinkovaný súbor – jedná sa o prílohu, ktorá má odkaz na umiestnenie v zariadení, kde bola pridaná a v rámci aplikácie Anzora teda nie je prístupná.

Ďalšou operáciou spojenou s prácou s položkami je manuálne vytváranie nových položiek. Pri zvolení tejto operácie sa zobrazí dialóg s voľbou typu položky. Novovytvorená položka je následne priradená do aktuálne vybranej kolekcie.

4.3. Vytvorenie webovej položky

Položky do knižnice je možné vytvárať aj prostredníctvom iných aplikácií. Konkrétne sa jedná o zdieľanie webovej stránky z aplikácie webového prehliadača. Princíp takéhoto prenosu dát medzi aplikáciami je bežným spôsobom v rámci platformy Android. Pri tejto operácii je zobrazený zoznam aplikácií podporujúci zdieľanie. V tomto zozname sa nachádza aj aplikácia Anzora, ktorá sa spustí po kliknutí na jej ikonku. Po spustení sa automaticky vytvorí nová položka typu webová stránka, pridá

sa do kolekcie „Nezaradené“. Následne sa otvorí obrazovka s jej informáciami, kde sa automaticky prevezme adresa URL a titulok webovej stránky ako názov položky.

4.4. Synchronizácia

Poslednou z funkcií je synchronizácia lokálnej a vzdialenej knižnice. Užívateľ spustí proces synchronizácie a na pozadí aplikácie sa začnú spracovávať zmeny. Najskôr sa aplikujú zmeny uložené na serveri a následne sú na server odoslané lokálne zmeny. Po dokončení procesu sa zobrazí informačný nápis „Synchronizácia ukončená“. Ak pri procese došlo u niektorých položiek ku konfliktu (položka bola aktualizovaná na serveri a zároveň upravená lokálne), po ukončení synchronizácie sa zobrazia ich názvy.

5. Popis implementácie

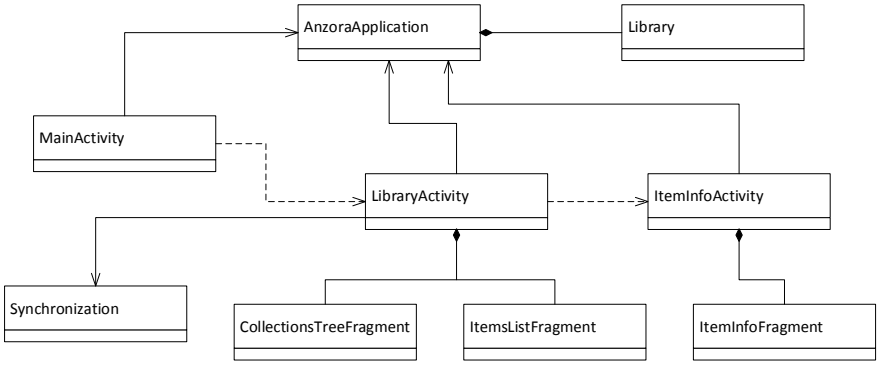
V tejto časti si popíšeme implementáciu aplikácie. Aby bol popis názornejší, pridali sme k jednotlivým častiam diagramy jednotlivých tried a v texte sme využili nasledujúce značenie:

- *NazovTriedy* – ak je použitá italika a slovo sa začína veľkým písmenom, ide o názov triedy obsiahnutej v platforme Android alebo v použitej knižnici.
- **NazovTriedy** – ak je slovo vyznačené tučným písmom a začína sa veľkým písmenom, ide o názov nami vytvorenej triedy.
- **inštancia** – ak je slovo vyznačené tučným písmom a začína sa malým písmenom, ide o názov inštančnej premennej.
- *metóda()* – ak je použitá italika, slovo sa začína malým písmenom a končí zátvorkami, ide o názov metódy.

5.1. Použité technológie

V rámci implementácie aplikácie sme využili štandardné technológie bežne používané pri vývoji Android aplikácií. Aplikácia je napísaná v jazyku Java vo verzii 1.8. Ďalej sme využili balíček org.json [9] pre spracovanie dát v JSON formáte a oauth.signpost [10] pre mechanizmus autentifikácie. Grafické prostredie je popísané pomocou XML s použitím štandardných Android UI komponent. Jedinú výnimku tvorí rozbaľovací strom, na ktorý sme použili knižnicu AndroidTreeView [11], a nástroj pre animáciu odstraňovania údajov zo zoznamov [12].

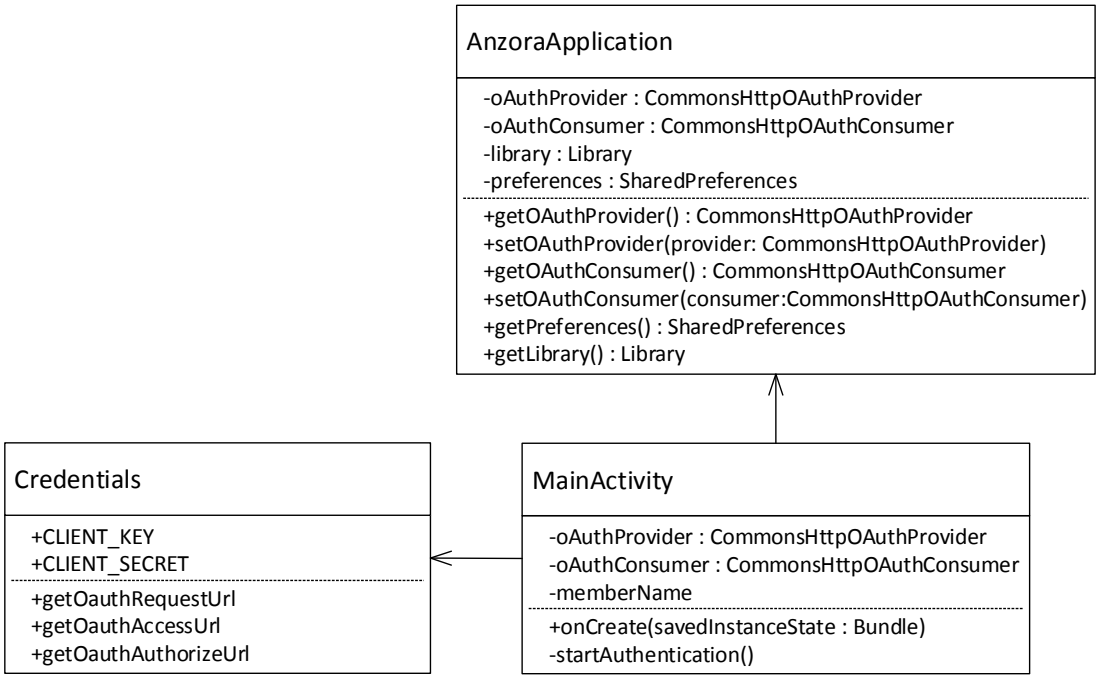
5.2. Základ aplikácie



Obrázok 5: Diagram základu aplikácie

Aplikácia je rozdelená na niekoľko hlavných tried (Obrázok 5), ktoré zabezpečujú jej funkcionalitu. Inštanciu celej aplikácie tvorí trieda **AnzoraApplication**, ktorá je potomkom triedy *Application*. Pre samotné fungovanie aplikácie a prechod medzi jednotlivými obrazovkami sú vytvorené triedy **MainActivity**, **LibraryActivity** a **ItemInfoActivity**, pričom všetky sú podedené od triedy *AppCompatActivity*. Na tieto objekty sú naviazané objekty podedené od triedy *Fragment* obsluhujúce užívateľské prostredie a jeho grafické komponenty. Na objekt **LibraryActivity** sú naviazané dva fragmenty **CollectionsTreeFragment** a **ItemsListFragment** a na **ItemInfoActivity** jeden fragment s názvom **ItemInfoFragment**. Pre správu lokálnej SQLite databázy je vytvorená trieda **Library.** Poslednou, hlavnou triedou, je trieda **Synchronization** podedená od *AsyncTask* zabezpečujúca proces synchronizácie. Okrem spomenutých tried sú implementované iné pomocné triedy, ktoré si popíšeme v ďalších častiach.

5.3. AnzoraApplication



Obrázok 6: Diagram AnzoraApplication a MainActivity

Hlavným objektom aplikácie je **AnzoraApplication** (Obrázok 6), ktorej úlohou je udržiavanie inštancií určitých objektov. Je to potrebné buď kvôli zachovaniu ich stavu pri prechode medzi konfiguračnými zmenami aplikácie, alebo prístupu k nim

v rámci celej aplikácie. K tomu nám pomôže to, že tento „singleton“ objekt žije po celú dobu behu aplikácie a jeho inštanciu je možné jednoducho získať v rámci ktorejkoľvek aktivity alebo fragmentu.

Jeho prvou dôležitou časťou sú objekty **oAuthProvider** a **oAuthConsumer**, ktoré uskutočňujú proces autentifikácie. Pri spustení aplikácie a začatí operácie prihlasovania užívateľa do Zotero online databázy sú tieto objekty inicializované. Následne, po spustení prihlasovania, je otvorený webový prehliadač a aplikácia je presunutá na pozadie a inštancie aktivít sú zničené. Po úspešnom prihlásení sa aktivity znovu načítajú a keďže inštancia objektu **AnzoraApplication** zničená nebola, je tak možné autentifikáciu dokončiť pôvodnými objektami **oAuthProvider** a **oAuthConsumer**.

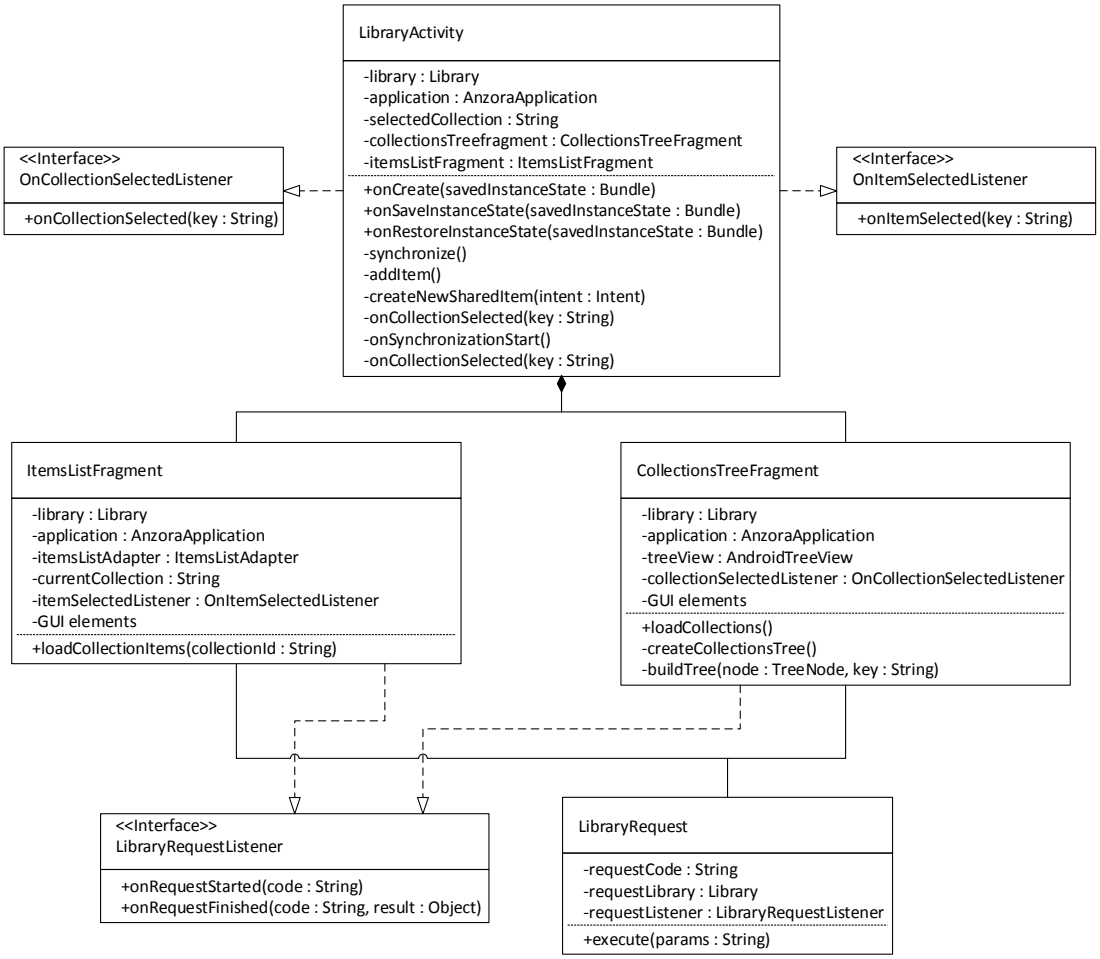
Druhou časťou sú objekty **library** a **preferences**, ku ktorým potrebujeme pristupovať z rôznych častí aplikácie. Ich inštancie sú vytvorené pri spustení aplikácie a následne sú sprístupnené pomocou metód *getLibrary()* a *getPreferences()*.

5.4. MainActivity

Aktivita **MainActivity** (Obrázok 6) je určená ako spúšťacia a jej úlohou je vykonanie procesu autentifikácie. Pri načítaní vytvorí hlavnú obrazovku s grafickými komponentami a získa inštancie autentifikačných objektov z **AnzoraApplication**. Keďže hlavná obrazovka nemá zložité užívateľské prostredie (obsahuje jednu funkčnú komponentu – tlačidlo) ani komponenty v „toolbare“, rozhodli sme sa použiť túto aktivitu bez fragmentu a všetky grafické komponenty z obrazovky vložiť do nej. Na spomínané tlačidlo je na udalosť stlačenia zaregistrovaná metóda *startAuthentication()*, ktorá spustí autentifikáciu. V rámci tejto metódy je vykonaný prvý bod autentifikácie, ako je popísaná v časti 2.2. Inicializujú sa autentifikačné objekty pomocou triedy **Credentials**, ktorá obsahuje statické konštanty s prístupovým kľúčom (CLIENT_KEY) a heslom (CLIENT_SECRET) pre povolenie aplikácie Anzora na prístup do Zotero databázy a adresy URL na overovací server. Po návrate z webového prehliadača je vykonaný tretí bod autentifikácie a prihlasovacie údaje pre užívateľa sa uložia do objektu **preferences** získaného z **AnzoraApplication**. Následne je vytvorený objekt triedy *Intent,* pomocou ktorého sa spustí aktivita **LibraryActivity.** Pri každom ďalšom spustení aplikácie sa pri vytváraní **MainActivity**

overí, či **preferences** obsahuje prihlasovacie údaje užívateľa. Ak áno, obrazovka **MainActivity** sa nezobrazí a spustí sa priamo **LibraryActivity.**

5.5. LibraryActivity



Obrázok 7: Diagram LibraryActivity

Táto aktivita (Obrázok 7) má za úlohu obsluhovať obrazovku pre prístup do užívateľskej knižnice. Keďže je táto obrazovka rozdelená na dve časti, rozhodli sme sa delegovať ich obsluhu do dvoch fragmentov. Aktivita tak poskytuje miesto pre vloženie týchto fragmentov. Sú to objekty **collectionsTreeFragment** a **itemsListFragment**.

Prvý uvedený fragment obsluhuje ľavú časť obrazovky s rozbaľovacím stromom obsahujúcim zoznam kolekcií v knižnici. Pri vytvorení tohto fragmentu pomocou metódy *onCreate()* sa získa z **AnzoraApplication** inštancia knižnice **library**, na ktorú sa odošle požiadavka o zoznam kolekcií. Každá kolekcia (inštancia triedy **Collection**) má svoj unikátny kľúč v rámci knižnice. Zároveň obsahuje záznam o tom,

ktorá kolekcia je jej rodičom (ak má rodiča je to jeho kľúč, ak nie, je to slovo „false“). Na základe toho je rekurzívnou DFS metódou *buildTree()* postavený strom kolekcií. Tento strom je vytvorený použitím *AndroidTreeView* knižnice a jej objektu *TreeNode.* Tento fragment zároveň obsahuje definíciu rozhrania **OnCollectionSelectedListener** a objekt **collectionSelectedListener**, ktorý toto rozhranie implementuje a reaguje tak na vybratie kolekcie užívateľom. Týmto objektom bude práve **LibraryActivity**, ktorý si kľúč danej kolekcie uloží.

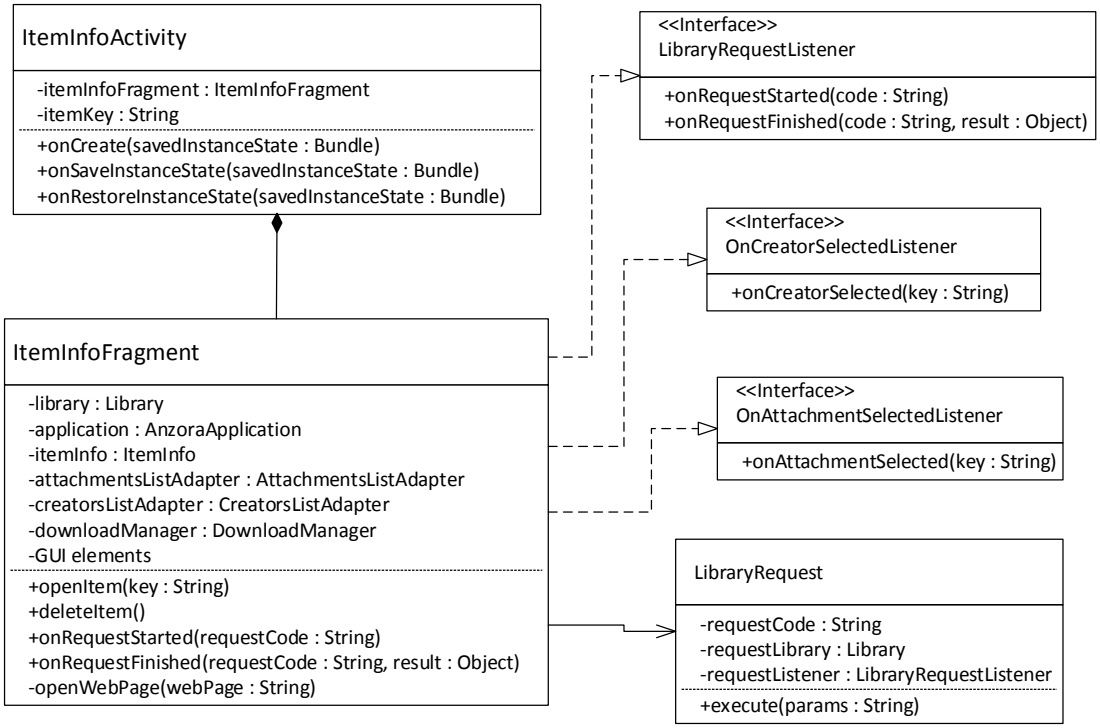
Druhý fragment obsluhuje pravú časť obrazovky so zoznamom položiek v aktuálne zvolenej kolekcii. Podobne, ako pri prvom fragmente, aj tu sa pri jeho vytvorení získa inštancia knižnice **library**. **ItemsListFragment** obsahuje verejnú metódu *loadCollectionItems()*, ktorá prijíma jeden parameter s kľúčom kolekcie. Jej zavolaním sa odošle požiadavka na knižnicu o zoznam položiek zo zvolenej kolekcie. Na udržiavanie tohto zoznamu slúži **ItemsListAdapter**. Fragment obsahuje definíciu rozhrania **OnItemSelectedListener** a objekt **itemSelectedListener**, ktorým bude znova **LibraryActivity** reagovať na vybratie položky užívateľom.

Celkovo teda implementovaním uvedených rozhraní poskytuje **LibraryActivity** mechanizmus komunikácie medzi fragmentami. Po zvolení kolekcie sa na **LibraryActivity** zavolá metóda *onCollectionSelected()* s daným kľúčom a následne sa s týmto kľúčom na **ItemsListFragment** zavolá *loadCollectionItems()*. Po zvolení položky sa s jej kľúčom na **LibraryActivity** zavolá metóda *onItemSelected()* a vytvorí sa nový objekt triedy *Intent* (kde sa kľúč vloží), ktorý spustí aktivitu **ItemInfoActivity**.

Ďalšou funkciou, ktorú aktivita obsluhuje, je vytvorenie novej položky. Pomocou metódy *addItem()* sa otvorí dialógové okno s výberom typu novej položky, kde po vybratí typu je spustená aktivita **ItemInfoActivity**. **LibraryActivity** je zároveň implementovaná tak, že reaguje na vytvorenie novej položky prostredníctvom zdieľania webovej stránky z webového prehliadača (napr. Chrome) zavolaním metódy *createNewSharedItem()*.

Poslednou obsluhovanou funkciou tejto aktivity je proces synchronizácie, ktorý sa spustí zavolaním metódy *synchronize()*. Operácia synchronizácie sa spúšťa ako asynchrónny proces a teda nie je presne dané, kedy skončí. Vytvorili sme mechanizmus „synchronization listenera“ pomocou rozhrania **SynchronizationListener** (popísaný v kapitole 5.8.). Toto rozhranie **LibraryActivity** implementuje a môže tak jednoducho na asynchrónny proces reagovať.

5.6. ItemInfoActivity



Obrázok 8: Diagram ItemInfoActivity

Poslednú obrazovku užívateľského rozhrania obsluhuje trieda **ItemInfoActivity** (Obrázok 8). Jej hlavnou úlohou je držiavať inštanciu fragmentu **ItemInfoFragment**, ktorý zabezpečuje logiku pre obrazovku. Aktivita reaguje jedine na „toolbar“ a jeho tlačidlá pre zmazanie položky a návrat na obrazovku s knižnicou. Fragment obsiahnutý v tejto aktivite obsahuje verejné metódy *openItem()* pre načítanie informácií o položke a *deleteItem()* pre jej zmazanie z knižnice. Po vybratí konkrétnej položky v obrazovke s knižnicou sa spustí **ItemInfoActivity** a aktivita dostane kľúč danej položky, ktorý potom odovzdá fragmentu zavolaním *openItem()*. Keďže aplikácia musí reagovať na otáčanie obrazovky, museli sme vymyslieť spôsob, akým na túto udalosť budeme reagovať v tejto aktivite. Trieda *AppCompatActivity,* od ktorej je **ItemInfoActivity** podedená obsahuje metódy, ktoré obsluhujú životný cyklus aktivity. Pri otočení displeja sa prostredie aktivity zničí a je zavolaná metóda *onSaveInstanceState(),* ktorá ako parameter dostane objekt triedy *Bundle*. Ten sa dá využiť na uloženie dát napríklad v podobe „stringu“. My sme to využili a do daného objektu sme vložili kľúč zobrazenej položky. Po otočení displeja sa prostredie aktivi-

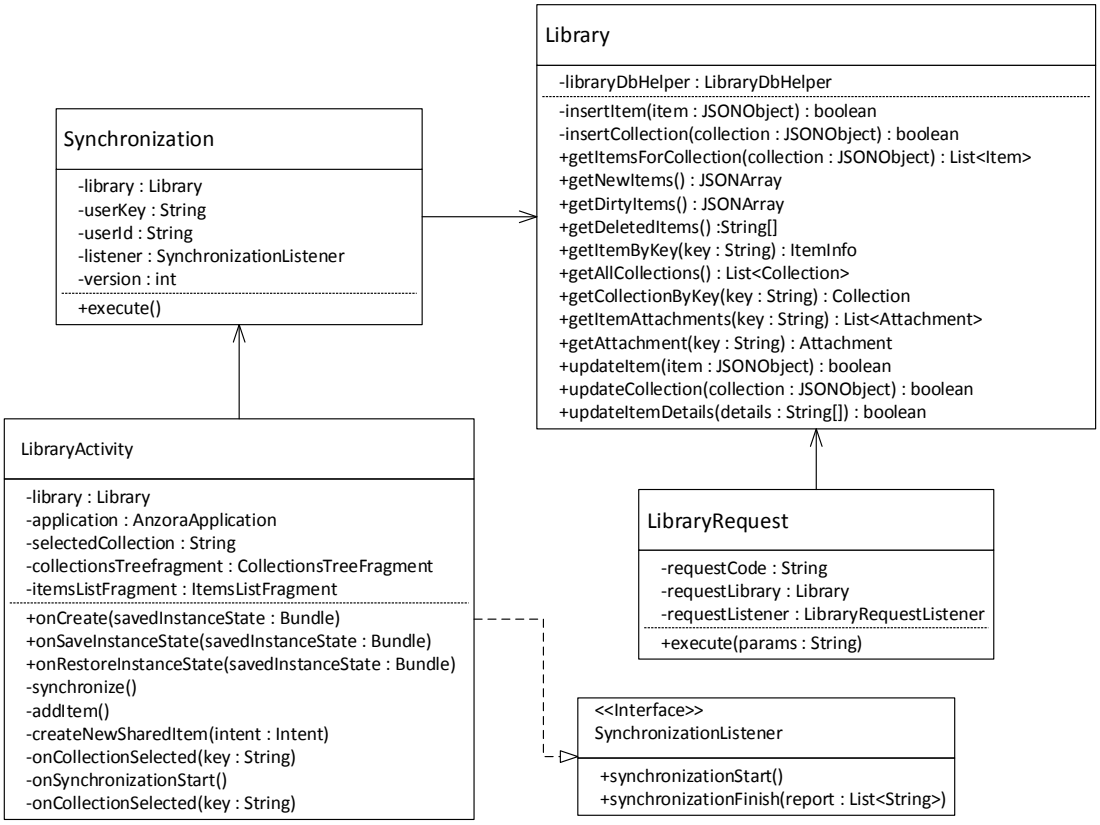
ty znova vytvorí, pričom sa zavolá metóda *onRestoreInstanceState()*, kde sa získa znova rovnaký objekt ako pri *onSaveInstanceState()*. Z neho získame kľúč, ktorý sme si tam uložili a zavoláme *openItem()*.

Ako bolo uvedené vyššie, **ItemInfoFragment** obsluhuje hlavnú časť obrazovky tejto aktivity. Pri jeho načítaní metódou *onCreate()* sa uloží do premennej **library** referencia na inštanciu databázy **Library** udržiavanú v **AnzoraApplication**. Prostredníctvom tejto referencie budeme pristupovať k dátam v lokálnej databáze. Dáta o konkrétnej položke sa do fragmentu načítajú zavolaním metódy *openItem()* z aktivity **ItemInfoActivity**, ktorá drží kľúč otváranej položky. V tejto metóde sa vytvorí žiadosť o dáta **LibraryRequest** a následne sa táto žiadosť spustí. Po dokončení fragment dostane všetky dáta danej položky ako objekt triedy **ItemInfo**.

Fragment obsahuje inštancie grafických komponent, ktoré využíva na zobrazovanie informácií o položke uložených v objekte triedy **ItemInfo**. Použité sú jednoduché textové komponenty vytvorené pomocou tried *EditText* a *TextView*. Pre zobrazenie zoznamov tvorcov a zoznamov príloh sme využili grafické prvky vytvorené pomocou triedy *RecyclerView*. Tento komponent bol vytvorený ako pokročilejšia verzia klasického *ListView*. Podľa dokumentácie je navrhnutý pre prípady, kedy sa zobrazujú kolekcie dát, ktorých elementy sa dynamicky menia za behu na základe interakcie užívateľa. To je práve prípad nášho zoznamu s tvorcami, ktorý užívateľ mení ich pridaním alebo odobratím. Zároveň tento komponent podporuje animácie, čo sme využili pre operáciu odobratia konkrétneho tvorcu jeho posunutím do strany. Pre udržiavanie spomínaných zoznamov sú použité adaptéry **CreatorsListAdapter** pre tvorcov a **AttachmentsListAdapter** pre prílohy. Pre reagovanie na kliknutie na prvky v týchto zoznamoch fragment implementuje dva rozhrania. Prvým je **OnAttachmentSelectedListener**, ktorý reaguje na kliknutie na prílohu. Po vybratí danej prílohy sa na pozadí aplikácie spustí proces sťahovania pomocou systémovej služby *DownloadManager*. Po dokončení je príloha uložená do interného úložiska zariadenia. Ak je príloha dostupná lokálne, po kliknutí na ňu sa otvorí použitím dostupných aplikácií nainštalovaných na zariadení. Druhým rozhraním je **OnCreatorSelectedListener,** ktorý reaguje na vybratie tvorcu metódou *onCreatorSelected()*. Po jej spustení sa vytvorí dialógové okno, ktoré umožňuje upravovať údaje o danom tvorcovi. Ďalšou pridanou funkciou fragmentu je sledovanie úpravy informácií o položke. Pomocou triedy *TextWatcher* sme vytvorili objekt, ktorý sleduje zmeny dát v textových grafických komponentoch. Ak dôjde k editácii, zmeníme časovú

značku pre čas a dátum zmeny, ktorá je súčasťou informácií o položke. Zároveň vieme, že pokiaľ došlo k zmene, položku musíme uložiť do databázy so značkou „dirty“, ktorá nám poslúži pri procese synchronizácie. Položka je ukladaná do databázy vždy, keď užívateľ odíde z obrazovky, napríklad návratom do obrazovky s knižnicou. Ušetrili sme tak užívateľa manuálneho potvrdzovania ukladania aktualizovaných informácii o položke a zároveň sme sa týmto priblížili „desktopovej“ verzii, kde sú všetky úpravy položky automaticky ukladané pri ich zmene. Poslednou funkciou fragmentu je otvorenie adresy URL, ktorá je súčasťou informácií zobrazenej na obrazovke. Po kliknutí na príslušné tlačidlo sa v metóde *openWebPage()* overí, či je URL validným odkazom na webovú stránku a následne sa vytvorí nová inštancia triedy *Intent,* ktorá spustí aplikáciu s webovým prehliadačom.

5.7. Library



Obrázok 9: Diagram Library a Synchronization

Trieda **Library** (Obrázok 9) slúži na obsluhu lokálnej knižnice. Ako už bolo uvedené, databáza, kde je knižnica uložená, je postavená na základe relačného databázového systému SQLite vstavaného priamo do systému Android. Pre implementáciu tejto

databázy sme postupovali podľa doporučených postupov zverejnených na oficiálnych Android „developer“ webových stránkach [13].

Prvým krokom bolo vytvorenie formálnej deklarácie organizácie databázy. Pre definíciu každej tabuľky sme použili pomocné triedy, ktoré pomocou konštánt určili názvy tabuliek a názvy stĺpcov. Tieto triedy sú vytvorené implementovaním rozhrania *BaseColumns*, ktoré už obsahuje definíciu pre primárny kľúč nazvaný _ID. Tento spôsob definovania tabuliek nám následne poskytoval veľkú výhodu hlavne pri vytváraní SQL dotazov, kde sme tieto konštanty použili. Tento príklad definuje konštanty pre tabuľku s informáciami o kolekciách v knižnici.

```
private static class CollectionsEntry implements BaseColumns {
    private static final String TABLE_NAME = "collections";
    private static final String COLUMN_KEY = "key";
    private static final String COLUMN_NAME = "name";
    private static final String COLUMN_VERSION = "version";
    private static final String COLUMN_PARENT = "parent";
}
```

Po tom, ako sme nadefinovali štruktúru databázy, bolo ďalším krokom implementovanie metód pre jej vytvorenie a údržbu. Príkladom je nasledujúci SQL príkaz na vytvorenie tabuľky pre kolekcie, kde vidíme princíp využitia nadefinovaných konštánt.

```
private static final String SQL_CREATE_COLLECTIONS =
        "CREATE TABLE " + CollectionsEntry.TABLE_NAME + " (" +
                CollectionsEntry._ID + " INTEGER PRIMARY KEY
AUTOINCREMENT, " +
                CollectionsEntry.COLUMN_KEY + " TEXT, " +
                CollectionsEntry.COLUMN_NAME + " TEXT NOT NULL, " +
                CollectionsEntry.COLUMN_VERSION + " INTEGER, " +
                CollectionsEntry.COLUMN_PARENT + " TEXT)";
```

Pre získanie referencie na našu databázu sme vytvorili privátnu triedu **LibraryDbHelper** dedením od triedy *SQLiteOpenHelper* (použitím jej metód *getWritableDatabase()* a *getReadableDatabase()*, ktoré otvoria databázu pre zápis a čítanie). Pre každú z operácií nad databázou, ktoré v rámci funkcionality aplikácie budeme využívať, sme implementovali samostatné metódy. Tieto metódy majú jednotnú postupnosť krokov:

1. Otvorenie databázy pomocou **LibraryDbHelper.**

2. Vytvorenie SQL dotazu pre danú operáciu (napríklad vloženie novej položky/kolekcie, zmazanie položky, aplikovanie zmien atď.)
3. Vykonanie dotazu a spracovanie jeho výsledkov.

Týmto spôsobom sme teda dostali celkový mechanizmus lokálnej knižnice.

Posledným krokom bolo vytvorenie spôsobu, akým budeme operácie spúšťať z užívateľského prostredia. Podmienkou použitia týchto operácií bolo, že musia bežať na pozadí aby nezaťažovali vlákno vyhradené pre samotnú aplikáciu. Použitím návrhového vzoru Command sme vytvorili triedu **LibraryRequest** podedenú od triedy *AsyncTask*, ktorá bude zapuzdrovať jednotlivé operácie nad databázou. Tieto operácie sa budú vykonávať na základe žiadosti. Fragmenty, ktoré s databázou komunikujú (**CollectionsTreeFragment**, **ItemsListFragment**, **ItemInfoFragment**), implementujú rozhranie **LibraryRequestListener**, ktoré obsahuje metódy *onRequestStarted()* (volaná pri začatí vykonávania žiadosti) a *onRequestFinished()* (volaná pri dokončení). Ak chce teda fragment napríklad získať dáta z knižnice, vytvorí novú inštanciu žiadosti, ktorá bude obsahovať typ žiadosti, inštanciu databázy a referenciu na samého seba a následne ju spustí s dodatočnými parametrami podľa jej typu. Príklad žiadosti o všetky kolekcie v knižnici volaný v **CollectionsTreeFragment**:

```
LibraryRequest request = new
LibraryRequest(LibraryRequest.GET_ALL_COLLECTIONS_REQ, library, this);
request.execute();
```

Po spustení žiadosti sa na základe jej typu zavolá prislúchajúca metóda na **Library** a výsledok tejto operácie je vrátený počúvajúcemu fragmentu. Pri sťahovaní dát zo serveru dostane trieda Library dáta v podobe objektov *JSONObject*. Z týchto dát sa pomocou kľúčov definovaných v Zotero Web API vyberajú konkrétne informácie, ktoré sú pomocou SQL výrazu vložené do databázy. Aby nedochádzalo k chybám, vytvorili sme si triedu **JsonKeys**, ktorá obsahuje JSON kľúče ako konštanty. Pri nahrávaní dát na server sú z databázy získané informácie, znova sa vytvoria objekty triedy *JSONObject* a pošlú sa na server. Pre prácu s dátami v rámci aplikácie sme vytvorili tieto triedy: **Item**, **ItemInfo**, **Collection** a **Attachment**.

5.8. Synchronization

Poslednou z hlavných tried je **Synchronization** (Obrázok 9). Jej úlohou je riadiť proces synchronizácie. Podobne, ako trieda **LibraryRequest**, aj táto trieda využíva praktiku návrhového vzoru Command. Trieda je podedená od triedy *AsyncTask* a vykonávanie procesu sa uskutočňuje na pozadí aplikácie. Keďže sme potrebovali reagovať na spustenie a ukončenie tohto asynchrónneho procesu, vytvorili sme rozhranie **SynchronizationListener** s metódami *onSynchronizationStart()* a *onSynchronizationFinish()*. Synchronizácia sa spúšťa v rámci aktivity **LibraryActivity**, ktorá toto rozhranie implementuje a reaguje tak na daný proces. Po stlačení príslušného tlačidla sa vytvorí nová inštancia **Synchronization** triedy, ktorá dostane ako parameter odkaz na inštanciu databázy, odkaz na **LibraryActivity**, uživateľský kľúč a permanentný komunikačný kľúč (získané z objektu **preferences** z **AnzoraApplication**) a aktuálnu verziu lokálnej databázy. Počas synchronizácie sa komunikuje so Zotero serverom prostredníctvom GET, POST a DELETE https žiadostí. Keďže prostredníctvom nich pristupujeme k osobnej vzdialenej databáze, musia obsahovať uvedené kľúče. Zároveň v týchto žiadostiach využívame verziu lokálnej databázy a tak vždy dostaneme ako odpoveď len nové dáta. Príklad žiadosti o kľúče položiek, ktoré majú verziu vyššiu ako 50 a odpovede.

```
https://api.zotero.org/users/2807516/items?since=50&format=versions&key=4Hby1pnm7qwYiTVT8SRSim1J
```

Každá https žiadosť má v hlavičke odpovede uvedenú aktuálnu verziu celej knižnice na serveri rovnú najvyššej verzii položky, alebo kolekcie v knižnici. Verziu z hlavičky budeme pri synchronizácii spracovávať. Pri implementácii algoritmu pre proces synchronizácie sme sa riadili postupom navrhnutým v dokumentácii Zotero WebApi s určitými úpravami špecifickými pre našu aplikáciu. Navrhli sme nasledujúci postup pozostávajúci z 5 častí:

1. Stiahnuť zoznam s kľúčmi a verziami položiek, kolekcií a zahodených položiek, ktoré sú na serveri uložené s verziou vyššou, ako je verzia lokálnej databázy. Pre každý objekt porovnať jeho verziu s lokálnou verziou. Ak sú rozdielne, pridať kľúč objektu do fronty pre stiahnutie. Vytvoriť https žiadosť pre nové dáta s použitím kľúčov z fronty. Táto žiadosť môže obsahovať maximálne 50 kľúčov. Ak ich je vo fronte viac ako 50, postupne posielať žiados-

ti po 50 kľúčoch a spracúvať odpovede. Odpovede spracovať rôzne podľa toho, či obsahuje aktualizované položky, kolekcie alebo položky z koša. Pre zoznam aktualizovaných položiek aplikovať postup:

```
Pre každú aktualizovanú položku:
        ak neexistuje lokálne
                vložiť ju do databázy s príznakom „not dirty“
                continue
        ak nebola lokálne upravená (má príznak „not dirty“)
                prepísať jej lokálne dáta s aktualizovanými
        inak
                pridať názov položky do zoznamu s konflikatmi a prepísať jej
                lokálne dáta s aktualizovanými a nastaviť príznak „not dirty“
```

Pre zoznam aktualizovaných kolekcií aplikovať: pre každú kolekciu prepísať lokálne dáta s aktualizovanými (naša aplikácia nepodporuje lokálne upravovanie kolekcií preto nemusíme riešiť konflikty pri kolekciách).

Pre zoznam s položkami v koši aplikovať: pre každú položku zmazať jej dáta z lokálnej databázy.

Po skončení tohto kroku zapamätať verziu knižnice z hlavičky poslednej odpovede ako dočasnú verziu.

2. Stiahnuť zoznam kľúčov zmazaných položiek a kolekcií na serveri. Pre zoznam zmazaných položiek aplikovať:

```
Pre každú zmazanú položku:
        ak neexistuje lokálne
                continue
        ak nebola lokálne upravená (má príznak „not dirty“)
                zmazať z databázy
        inak
                pridať názov položky do zoznamu s konflikatmi a zmazať z
                databázy
```

Pre zoznam zmazaných kolekcií aplikovať: pre každú zmazanú kolekciu odstrániť z lokálnej databázy.

3. Poznačiť si verziu knižnice z hlavičky odpovede na žiadosť o zmazané objekty a porovnať ho s dočasnou verziou. Ak sú rozdielne (kým sme aplikovali zmeny knižnica na serveri bola znova aktualizovaná), vrátiť sa do bodu 1. inak pokračovať do bodu 4.
4. Poslať na server https POST žiadosti s lokálnymi zmenami (položky s príznakom „dirty“) a novými položkami (každá z týchto žiadostí musí znova obsahovať maximálne 50 položiek). Z hlavičky odpovede vybrať novú

verziu pre lokálnu knižnicu a zapamätať si ju do výsledku synchronizácie. Zároveň túto verziu nastaviť lokálnym položkám, ktorých aktualizácie sa odosielali na server.

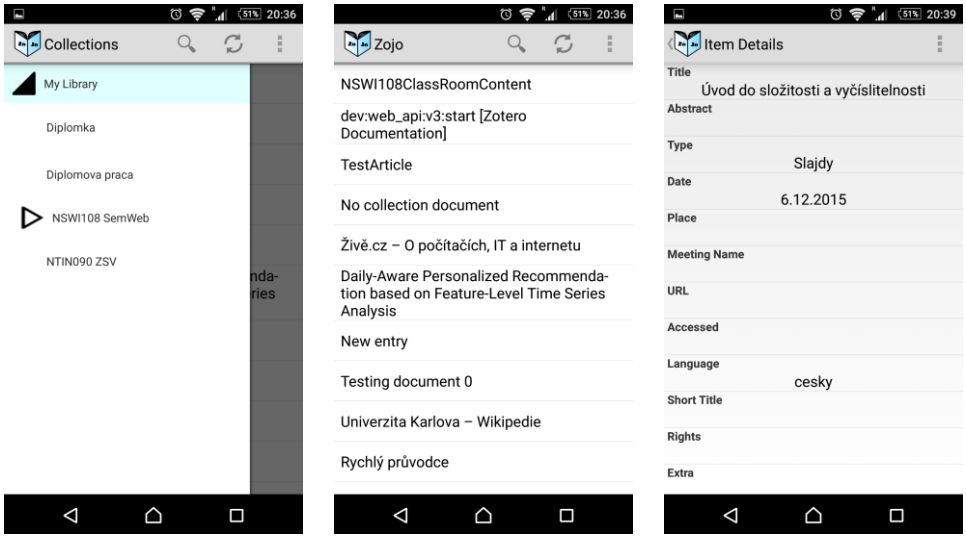
5. Nahrať na server kľúče lokálnych zmazaných položiek.

Tento postup je v triede **Synchronization** implementovaný ako proces na pozadí. Po jeho dokončení sa na počúvajúcom objekte (v našom prípade **LibraryActivity**) zavolá metóda *onSynchronizationFinish()*, ktorá dostane výsledok synchronizácie – novú verziu pre lokálnu knižnicu a zoznam konfliktov, ak k nejakým došlo. Verzia knižnice sa uloží do objektu s preferenciami aplikácie a bude použitá pri budúcej synchronizácii.

6. Porovnanie existujúcich aplikácií

Pre prácu so Zotero online knižnicou existuje v súčasnosti niekoľko mobilných aplikácií pre systém Android dostupných v službe Google Play. My sme ich otestovali a v tejto kapitole si ich stručne popíšeme a porovnáme ich funkcionalitu.

6.1. Zojo

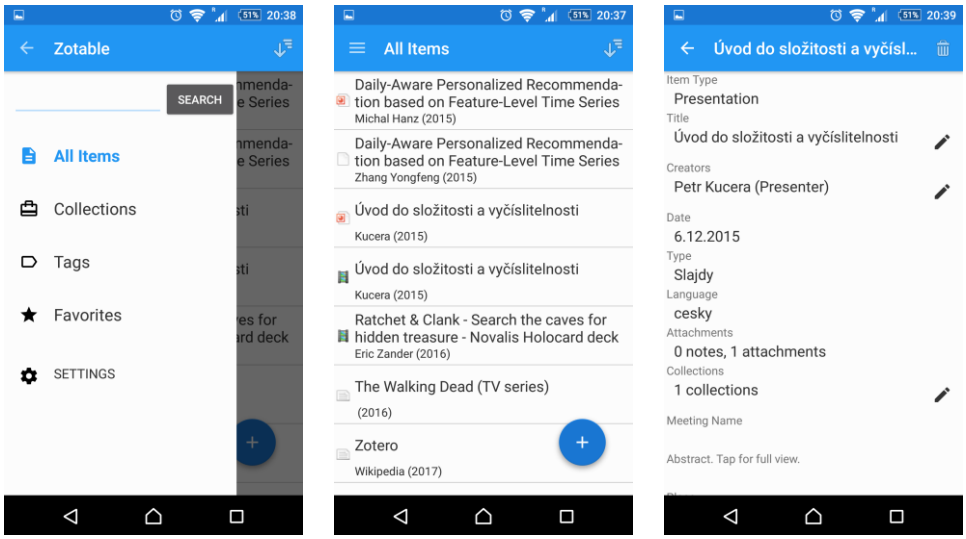


Obrázok 10: Zojo

Ako prvou testovanou aplikáciou bolo Zojo (Obrázok 10). Poskytuje prístup k osobnej knižnici po prihlásení užívateľa a synchronizácii s online serverom. Páčilo sa nám jednoduché užívateľské prostredie, kde má užívateľ zobrazený rozbaľovací strom s kolekciami. Po výbere danej kolekcie sa zobrazia položky do nej patriace. To však má nevýhodu v tom, že tento strom je zobrazený v postrannej vysúvacej časti obrazovky. Ak sme teda chceli prepínať medzi kolekciami, museli sme vždy otvoriť tento strom, vybrať kolekciu a znova ho zatvoriť. To nám prišlo nepraktické. Aplikácia však poskytuje zobrazenie položiek na základe vyhľadávania ich názvov. Po vybratí konkrétnej položky sa zobrazí prehľad všetkých informácií o danej položke. Určitým obmedzením však je, že tieto informácie nie je možné upravovať, ani nijak s nimi pracovať (napríklad otvoriť adresu URL pridanú k položke). Ďalšou veľkou nevýhodou je zobrazovanie príloh. V rámci ich sťahovania aplikácia nekomunikuje priamo so Zotero serverom ale s cloudovým úložiskom Dropbox. Na to, aby si užívateľ stiahol prílohy, musí mať vytvorený Dropbox účet. Ďalej musí na počítači zistiť, kam „desktopová“ verzia Zotera ukladá prílohy a tento adresár napojiť na Dropbox

úložisko a nahrať tam všetky prílohy. Následne aplikáciu Zojo pripojiť k Dropbox účtu a stiahnuť do mobilného zariadenia konkrétne prílohy. Tento postup je podľa nás dosť nepraktický a užívateľa skôr odradzuje od používania tejto aplikácie.

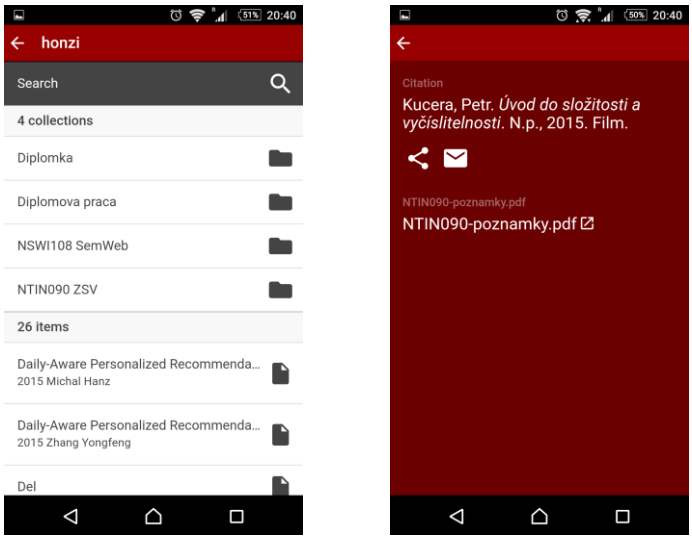
6.2. Zotable



Obrázok 11: Zotable

Ďalšou testovanou bola aplikácia Zotable (Obrázok 11). Tá rovnako, ako Zojo, poskytuje po prihlásení užívateľa prístup k jeho osobnej knižnici. Užívateľské prostredie je prehľadné a o jednotlivých položkách v knižnici zobrazuje všetky informácie. Zobrazenie položiek je možné vyhľadaním názvu alebo vybratím kolekcie. Na rozdiel od prvej testovanej aplikácie táto už neslúži len na zobrazovanie, ale poskytuje aj možnosť úpravy informácií o položkách. Konkrétne sa jedná o úpravu názvu položky, jej zoznamu tvorcov, zoznamu kolekcií, v ktorých sa položka nachádza a jej značky. Zároveň je možné pridávať nové položky manuálnym vyplnením informácií, automatickým vyplnením podľa dát z PDF súboru, naskenovaním čiarového kódu publikácie a vyhľadaním podľa ISBN kódu. Užitočnými funkciami aplikácie sú pridávanie kolekcií do zoznamu obľúbených a následne rýchlejší prístup k nim a otváranie adresy URL položky spustením webového prehliadača. Na druhú stranu aplikácia neposkytuje žiaden prístup k prílohám. Ak sme otvorili položku s prílohou, zobrazil sa jedine údaj o počte príloh bez možnosti ich stiahnutia. Tento fakt je veľkou nevýhodou aplikácie.

6.3. Zed Lite



Obrázok 12: Zed Lite

Poslednou testovanou bola aplikácia Zed Lite (Obrázok 12). Používanie tejto aplikácie bolo značne obmedzené. Po synchronizácii bolo možné prehliadať obsah knižnice a vyhľadávať položky podľa názvu. Informácie o položke sa zobrazili len vo formáte citácie a tú bolo možné zdieľať napríklad odoslaním prostredníctvom e-mailu. Zároveň sa dali stiahnuť jej prílohy a následne otvoriť. Aplikácia nevytvárala lokálnu databázu, a tak nebolo možné s ňou pracovať bez prístupu k internetovému pripojeniu. Všetky dáta sa len dočasne ukladali a pri jej každom spustení bolo nutné všetky dáta a prílohy znovu sťahovať. Podľa informácii uvedených v službe Google Play je táto aplikácia len zmenšenou verziou plnej aplikácie s názvom Zed. Nám sa ju však nepodarilo získať, keďže nebola dostupná v službe Google Play a taktiež webová stránka, na ktorú sa autor odkazoval, neexistovala.

7. Záver

V tejto práci sme navrhli a implementovali aplikáciu Anzora pre mobilné zariadenia so systémom Android, ktorá poskytuje prístup a správu k osobnej databáze zdrojov Zotero. Používaním existujúcich aplikácií sme zistili určité nedostatky, ktoré nás viedli k vytvoreniu vlastnej aplikácie. V našom riešení sme sa zamerali na to, aby bolo jednoduché na použitie a poskytovalo prístup k osobnej knižnici Zotero v „online“ aj „offline“ móde. Naša aplikácia by sa v budúcnosti dala rozšíriť o možnosť editovania kolekcií a pokročilejšie spracovanie synchronizačných konfliktov. Celkovo bola táto práca úspešná.

Zoznam použitej literatúry

1. **BibDesk.** BibDesk. [Online] [cit: 2017-05-07] http://bibdesk.sourceforge.net/.

2. **Digital Science & Research Solutions Inc.** ReadCube. [Online] [cit: 2017-05-07] https://www.readcube.com/.

3. **Digital Science & Research Solutions Inc.** Papers. [Online] [cit: 2017-05-07] http://papersapp.com/.

4. **Center for History and New Media.** Zotero. [Online] [cit: 2017-05-07] https://www.zotero.org/.

5. **Center for History and New Media.** Web API. [Online][cit: 2017-05-07] https://www.zotero.org/support/dev/web_api/v3/start.

6. **Center for History and New Media.** Zotero Download. [Online] [cit: 2017-05-07] https://www.zotero.org/download/.

7. **Wikipedia.** Atom (standard) - Wikipedia. [Online] [cit: 2017-05-07] https://en.wikipedia.org/wiki/Atom_(standard).

8. **Google.** Dashboards | Android Developers. [Online] [cit: 2017-05-06] https://developer.android.com/about/dashboards/index.html.

9. **Google.** org.json | Android Developers. [Online] [cit: 2017-05-07] https://developer.android.com/reference/org/json/package-summary.html.

10. **Käppler, Matthias.** GitHub - mttkay/signpost: A light-weight client-side OAuth library for Java. *GitHub.* [Online] [cit: 2017-05-07] https://github.com/mttkay/signpost.

11. **Melnychuk, Bogdan.** GitHub - bmelnychuk/AndroidTreeView: AndroidTreeView. TreeView implementation for android. *GitHub.* [Online] [cit: 2017-05-07] https://github.com/bmelnychuk/AndroidTreeView.

12. **Kovačevic, Nemanja.** Recycler-view-swipe-to-delete. [Online] [cit: 2017-05-07] https://github.com/nemanja-kovacevic/recycler-view-swipe-to-delete.

13. **Google.** Saving Data in SQL Databases | Android Developers. [Online] [cit: 2017-05-07] https://developer.android.com/training/basics/data-storage/databases.html.

14. **Media, Center for History and New.** Zotero | Register. [Online] [cit: 2017-05-08] https://www.zotero.org/user/register/.

8. Prílohy

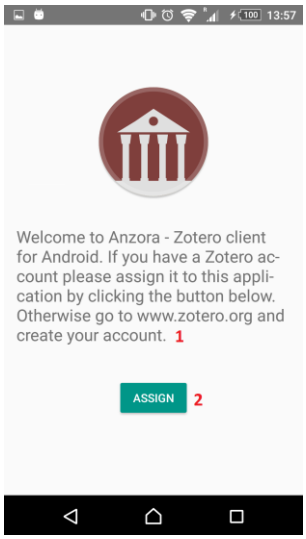
8.1. Užívateľská dokumentácia

Užívateľské prostredie aplikácie Anzora sa skladá z troch obrazoviek. Každá z týchto obrazoviek ma špecifickú úlohu v rámci práce s aplikáciou a obsluhuje jednotlivé funkcie. Sú to tieto obrazovky:

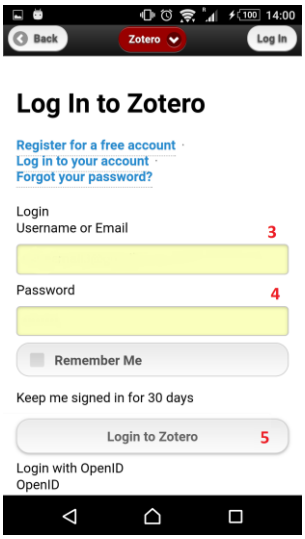
- prihlasovacia obrazovka
- obrazovka s obsahom knižnice
- obrazovka s informáciami o položke

Aplikácia, pre určité funkcie, využíva komunikáciu so Zotero serverom. Preto je potrebné, aby pre správne fungovanie týchto funkcií malo mobilné zariadenie stabilné pripojenie na internet, najlepšie pomocou WiFi pripojenia. Zároveň je nutné, aby mal užívateľ zaregistrovaný účet na Zotero serveri [14] a na tomto účte vytvorenú osobnú knižnicu, ktorú chce aplikáciou Anzora spravovať. V nasledujúcich kapitolách budeme pre odkazovanie na grafické komponenty očíslované v obrázkoch používať skratku „komp“ a číslo komponentu.

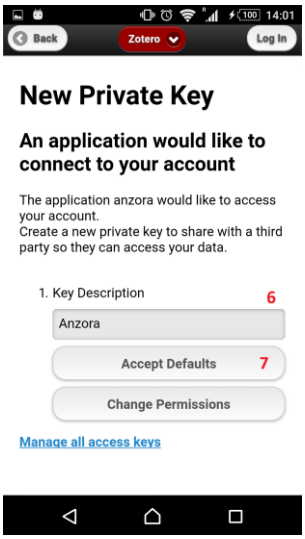
8.1.1. Prihlasovacia obrazovka



Obrázok 13



Obrázok 14

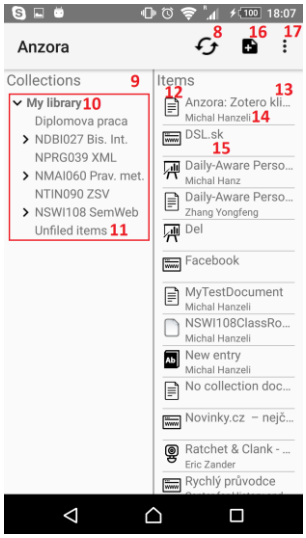


Obrázok 15

Prihlasovacia obrazovka (Obrázok 13) tvorí úvodnú obrazovku po prvotnom spustení aplikácie. Obsahuje logo aplikácie a uvítací text (komp. 1) so stručný-

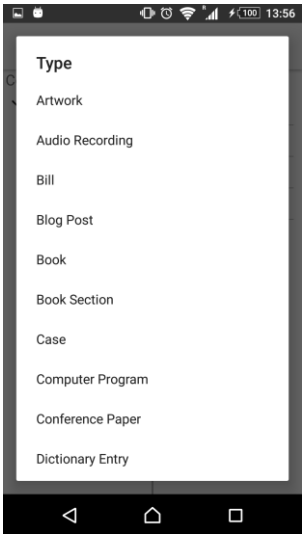
mi informáciami o tom, že ak má užívateľ účet na serveri Zotero, môže ho priradiť k aplikácii Anzora, inak si ho musí vytvoriť. Pod textom sa nachádza tlačidlo (komp. 2) a po jeho stlačení sa otvorí webový prehliadač s prihlasovacou stránkou serveru Zotero (Obrázok 14). Tu užívateľ vyplní svoje prihlasovacie meno do poľa (komp. 3) a heslo do poľa (komp. 4). Následne stlačí tlačidlo (komp. 5) a otvorí sa ďalšia stránka (Obrázok 15) s informáciou o tom, že aplikácia Anzora žiada o povolenie k prístupu do užívateľovej osobnej knižnice. V poli (komp. 6) sa zobrazí názov aplikácie a po stlačení tlačidla (komp. 7) sa znovu spustí Anzora. Je potrebné, aby užívateľ tlačidlom (komp. 7) potvrdil štandardné nastavenia, ktorými sa povolia práva na čítanie aj zapisovanie do knižnice. Po tomto procese prihlasovania sa zobrazí obrazovka s obsahom knižnice popísaná v nasledujúcej kapitole. Prihlasovacia obrazovka sa znova pri opätovnom spustení aplikácie nezobrazí až do chvíle, kým sa užívateľ nerozhodne manuálne odhlásiť.

8.1.2. Obrazovka s obsahom knižnice



Obrázok 16

Obrázok 17



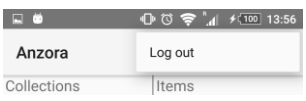
Obrázok 18

Táto obrazovka (Obrázok 16) slúži na prehliadanie obsahu knižnice. Do obrazovky sa užívateľ dostane po úspešnom prihlásení postupom popísaným v predošlej kapitole. Po prihlásení ešte knižnica neobsahuje žiadne kolekcie ani položky a je potrebné spustiť prvotnú synchronizáciu pomocou tlačidla (komp. 8) umiestneného v hornej časti obrazovky. Stlačením tohto tlačidla sa synchronizácia spustí a zobrazí informačný text oznamujúci začatie synchronizácie. Po jej skončení sa znovu zobrazí in-

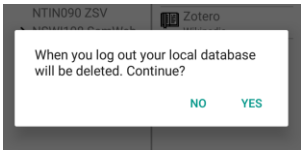
formačný text a knižnica sa automaticky obnoví. Aplikácia Anzora je navrhnutá tak, že proces synchronizácie je nutné spúšťať manuálne. Ak teda užívateľ vykoná úpravy v lokálnej knižnici a chce tieto zmeny nahrať na server, musí spustiť synchronizáciu stlačením príslušného tlačidla.

Po prvotnej synchronizácii sa v ľavej časti obrazovky zobrazí rozbaľovací strom (komp. 9), ktorý obsahuje názvy kolekcií v knižnici. V pravej časti obrazovky je zobrazený zoznam položiek. Pomocou stromu užívateľ prehliada knižnice nasledujúcim spôsobom. Kolekcie, ktoré obsahujú podkolekcie, majú vľavo od názvu zobrazenú šípku. Po kliknutí na túto šípku sa zobrazia podkolekcie patriace danej kolekcii. Ďalšími ovládateľnými prvkami v strome sú názvy kolekcií. Po kliknutí na názov sa tento názov zobrazí hrubým pásmom, pravá časť obrazovky sa obnoví a zobrazia sa položky patriace do zvolenej kolekcie (Obrázok 17). Ak je vybraný koreň stromu (komp. 10), v pravej časti sú zobrazené všetky položky v knižnici. Strom obsahuje ešte jednu špeciálnu kolekciu označenú „Unfiled items“ (komp. 11). Po kliknutí na túto kolekciu sa v zozname zobrazia položky, ktoré nie sú priradené do žiadnej kolekcie. Každá položka v zozname sa skladá z troch častí. Prvou je ikonka (komp. 12), ktorá označuje typ položky. Ďalej nasleduje názov položky (komp. 13). Treťou časťou je meno tvorcu (komp. 14), ktoré je zobrazené pod názvom položky (ak položka nemá tvorcu zobrazí sa prázdne miesto, komp. 15). Po kliknutí na konkrétnu položku sa otvorí obrazovka s informáciami o vybranej položke, popísaná v nasledujúcej kapitole.

Horná časť obrazovky obsahuje tlačidlo (komp. 16) pomocou ktorého užívateľ vytvorí novú položku do aktuálne vybranej kolekcie. Po stlačení tohto tlačidla sa zobrazí obrazovka (Obrázok 18), ktorá obsahuje zoznam s typmi položiek. Ak si užívateľ niektorý typ vyberie, vytvorí sa nová položka vybraného typu a otvorí sa rovnaká obrazovka ako v prípade výberu existujúcej položky, ale bez podrobných informácií.



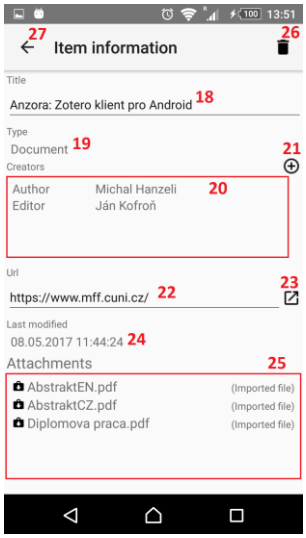
Obrázok 19



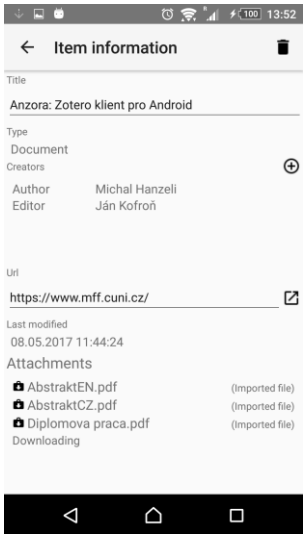
Obrázok 20

Poslednou operáciou, ktorú je možné vykonať na obrazovke s knižnicou, je odhlásenie užívateľa. Po stlačení tlačidla (komp. 17) sa v hornej časti obrazovky zobrazí menu s „Log out“ tlačidlom (Obrázok 19). Ak ho užívateľ stlačí, zobrazí sa dialóg (Obrázok 20) s informáciou o tom, že ak odhlásenie potvrdí lokálna knižnica bude zmazaná.

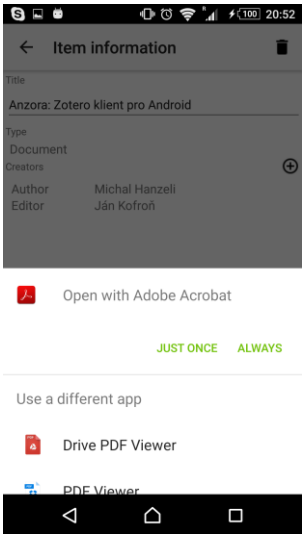
8.1.3. Obrazovka s informáciami o položke



Obrázok 21

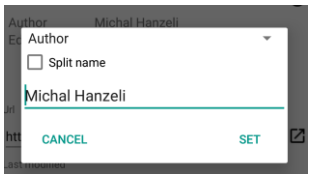


Obrázok 22

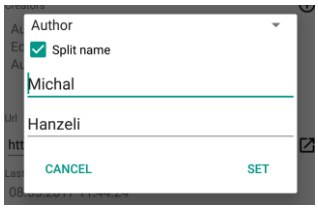


Obrázok 23

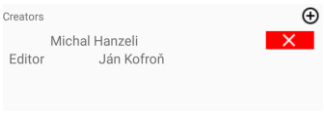
Poslednou, hlavnou časťou užívateľského prostredia, je obrazovka s informáciami o konkrétnej položke (Obrázok 21). Zobrazí sa ak užívateľ v obrazovke popísanej v predošlej kapitole vyberie niektorú položku zo zoznamu alebo vytvorí novú. Každá informácia zobrazená na tejto obrazovke obsahuje titulok, ktorý ju popisuje. O položke sú zobrazené tieto údaje: názov (komp. 18), typ (komp. 19), zoznam tvorcov (komp. 20), adresa URL (komp. 22), dátum a čas poslednej modifikácie (komp. 24) a zoznam príloh (komp. 25). Názov a adresa URL sú editovateľné polia a užívateľ tak môže tieto dáta upravovať. Prílohy sa skladajú z troch častí. Prvá je ikona, ktorá označuje, či je príloha stiahnutá na zariadení (💼) alebo nie (💼). Za ňou nasleduje názov súboru prílohy a ďalej text označujúci typ prílohy. Po zvolení prílohy sa začne príloha sťahovať (Obrázok 22), alebo ak už je lokálne dostupná zobrazí sa zoznam aplikácií, ktoré daný súbor dokáže otvoriť (Obrázok 23).



Obrázok 24



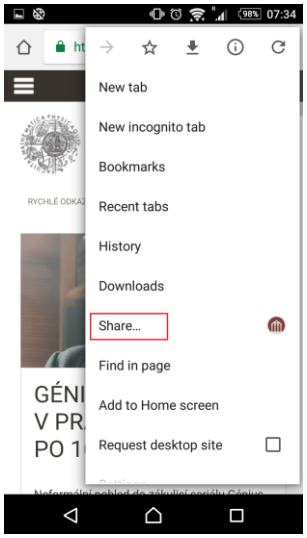
Obrázok 25



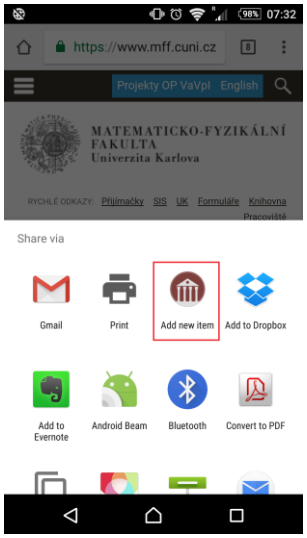
Obrázok 26

Stlačením riadku s tvorcom sa zobrazí dialóg (Obrázok 24), ktorý umožňuje upravovať údaje o danom tvorcovi – typ tvorcu, jeho meno a priezvisko. Pomocou zaškrtávajúceho tlačidla je možné rozdeliť meno a priezvisko do samostatných polí (Obrázok 25). Ak chce užívateľ zmazať tvorcu, potiahne daný riadok do ľavej strany, pričom sa zobrazí animácia odstraňovania (Obrázok 26). Pre pridávanie tvorcov slúži tlačidlo umiestnené nad zoznamom (komp. 21). Poslednými funkčnými prvkami v tejto obrazovke sú tlačidlo komp. 23 pre otvorenie adresy URL vo webovom prehliadači, tlačidlo komp. 26 pre zmazanie zobrazenej položky a tlačidlo komp. 27 v hornej lište pre návrat do obrazovky s obsahom knižnice.

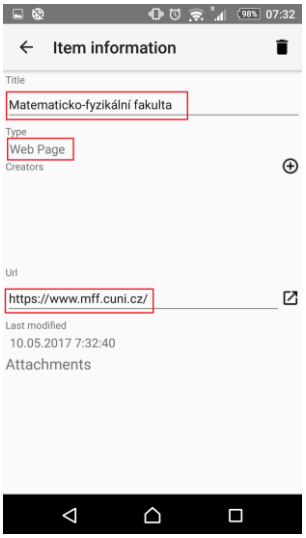
8.1.4. Zdieľanie webovej stránky



Obrázok 27



Obrázok 28



Obrázok 29

Aplikácia Anzora umožňuje vytvárať nové položky typu web stránka pomocou operácie zdieľania webovej stránky z prostredia mobilného webového prehliadača. V aplikácii webového prehliadača užívateľ otvorí menu a zvolí možnosť zdieľať (Obrázok 27). Následne sa otvorí zoznam aplikácii, ktoré dokážu zdieľané dáta spracovať. V tomto zozname sa bude nachádzať aj aplikácia Anzora (Obrázok 28). Po kliknutí na jej ikonku sa aplikácia otvorí do obrazovky s informáciami o položke kde

sa automaticky vyplní názov položky, typ a adresa URL (Obrázok 29). Táto nová položka sa uloží do knižnice bez pridelenia konkrétnej kolekcie.

8.2. Inštalácia aplikácie

Prílohou práce je prenosové médium, ktoré obsahuje inštalačný súbor s aplikáciou Anzora. Postup inštalácie je nasledovný:

1. V nastaveniach mobilného zariadenia povoliť inštalovanie aplikácií z **neznámych zdrojov**.
2. Nainštalovať na zariadenie súborový manažér, napr. **Total Commander** prostredníctvom služby **Google Play**.
3. Stiahnuť inštalačný balíček **Anzora.apk** do mobilného zariadenia.
4. V súborovom manažérovi otvoriť inštalačný balíček a po vyzvaní potvrdiť spustenie aplikácie
5. Po ukončení inštalácie sa aplikácia pridá do zoznamu všetkých nainštalovaných aplikácií odkiaľ je možné ju spustiť.

8.3. Obsah priloženého prenosového média

V priloženom prenosovom médiu sa nachádza:

- inštalačný balíček
- zdrojový kód aplikácie Anzora
- dokumentácia k zdrojovému kódu vygenerovaná pomocou Javadoc
- PDF verzia diplomovej práce